POLITECNICO
MILANO 1863

MSc in Computer Science and Engineering

Software Engineering 2 Project

TrackMe Requirement Analysis and Specification Document

Professor:
Elisabetta di Nitto

Authors :
Andrea Biscontini - 901310
Marco Gelli - 901470
Alvise de'Faveri Tron - 920882

Contents

1 Introduction

1.1 Purpose

This document is the Requirement Analysis and Specification Document of the Data4Help system and the two subsystems working on top of it: AutomatedSOS and Track4Run. In this document we will describe the general purpose of the system, the main goals that it has to achieve, the functional and non-functional requirements that have to be met and the domain assumptions and constraints that have been identified.

This document is addressed to all the stakeholders of the system: customers (end users or third parties), system analysts, project managers and developers. Here a brief overview of the product will be given.

Data management is a very popular topic nowadays. It has applications in multiple fields and it may result in many benefits when properly employed: for example medical research, healthcare and the fitness world have greatly benefited from an extensive use of personal data in the past years. On the other hand, data treatment and security has been a increasingly important concern among our society, which resulted in a more strict and aware regulatory policies.

Data4Help is a system designed to address the issue of collecting user data from different sources like smart devices or external applications and ensure a secure and reliable way to share them with interested companies. Additionally, some other functionalities are added to the system by AutomatedSOS and Track4Run.

The main purpose of AutomatedSOS is to offer to Data4Help users an automated solution to monitor their health status and call an ambulance that will rescue them in case of emergency. This is done by exploiting the data collected by the Data4Help system.

Finally, Track4Run has the objective to solve the organizational problems that arise in the management of sport events, in particular running events that in our days are becoming more and more popular, even among amateurs. In fact, this system gives to each user the possibility to organize and participate to runs, providing also the possibility, for those who don't want to run, to watch the participants from the sideline.

1.2 Scope

1.2.1 Description of the given problem

As stated in the above section, Data4Help main goal is to collect user data and make them available to third parties, all while guaranteeing the user privacy and consensus in personal data processing.

To collect these data, the system needs to connect to users' smart devices and wearables so that they can send the retrieved information to the TrackMe proprietary system. This data are then processed by Data4Help and made available for third party organizations under certain conditions.

In particular, we can divide requests in two different types: single user data requests, that are forwarded directly to the individual, who can accept or refuse them, and data requests for groups of individuals, that are handled by Data4Help and are carried out only if there is the possibility to properly anonymize the data. Third parties could also desire to look for future changes in the data they requested: for this reason an auxiliary subscription to new data is also provided.

Moving to AutomatedSOS, it can be used to help elderly or sick people to monitor their health conditions and intervene in the case of an emergency. In fact, the goal of AutomatedSOS is to be very reactive (maximum 5 seconds) in detecting possible health problems and immediately call an ambulance to the user location.

Instead, Track4Run is designed for being used by a multitude of different users: any user can become the organizer of a run by creating one. The run creation procedure is made really simple: the user just needs to insert the information related to the event and select a route for the run on the map. When a run

is created, every other user can enroll to it. To give an even better service, there is also the possibility to follow every runner's position on a live GPS map.

1.2.2 Goals

- **[G1]** Data4Help must be able to keep track of real time health status and position from registered users
- **[G2]** Data4Help should allow third parties to gather information from a single user or from an anonymous group of users
- **[G3]** Data4Help should allow third parties to subscribe to new data and receive them as soon as they're produced
- **[G4]** AutomatedSOS should be able to identify an health emergency when the user data are below/exceeding a certain thresholds
- **[G5]** AutomatedSOS must call an ambulance when it detects a health emergency
- **[G7]** Track4Run must allow a user to create and manage running events
- **[G8]** Track4Run must allow a user to participate to an organized run
- **[G9]** Track4Run must allow a spectator to track the position of the participants of an ongoing run

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Third Party**: Any system that is not Data4Help that wants to access to Data4Help user's data
- **Data Request**: The action that third parties can make to retrieve users data
- **Notification**: An email, SMS or any other way in which a user can be alerted of any event of the system
- **Data Source**: An external device or application that can collect data from the user
- **Synchronization**: The procedure to connect a Data Source with the Data4Help system
- **Parameter**: A generic physical or health information regarding the user that can be collected from a device
- **Threshold**: A value defining when a certain parameter is out of the normal range
- **Emergency Status**: A status in which one or more parameters of an AutomatedSOS user exceeded their thresholds

1.3.2 Acronyms

- **RASD**: Requirement Analysis and Specification Document
- **API**: Application Programming Interface
- **GPS**: Global Positioning System
- **GDPR**: General Data Protection Regulation

1.3.3 Abbreviations

- **Gn**: n-th goal
- **Dn**: n-th domain assumption
- **Rn**: n-th functional requirement

1.4 Revision history

Date	Applied Changes
11/11/2018	First issue of the document
13/11/2018	Added Sequence Diagrams images in the Functional Requirements section. Added Mockup images in the External Interfaces section. Added Alloy text and images in the Alloy section. Fixed use case tables and corresponding order. Fixed numbers of Requirements and Domain Assumptions in the Functional Requirements section. Added Revision History in the introduction.
10/12/2018	Removed R18 as it is not needed for the system to function. Also removed third party mockups.

1.5 Document Structure

Chapter 1 gives an introduction to the problem and describes the goals and purpose of the application.

Chapter 2 presents an overall description of the system. Firstly a description of the domain model and shared phenomena are provided coupled with detailed UML diagrams. Secondly the major functions of the system are more precisely specified and connected with the goals and domain constraints of the system.

Chapter 3 contains all the requirements that the system must meet in order to accomplish the given goals. In particular mockups of the user interfaces, use cases and sequence diagrams are presented to give a complete description of what features the system should and should not offer.

Chapter 4 includes the Alloy model with an example of a generated world.

Chapter 5 shows the effort by each group member.

2 Overall Description

2.1 Product perspective

Our product is an ecosystem of applications that are designed from scratch to accomplish different purposes.

In particular, the system is designed to be a set of three subsystems interacting with each other, as shown in the following diagram:

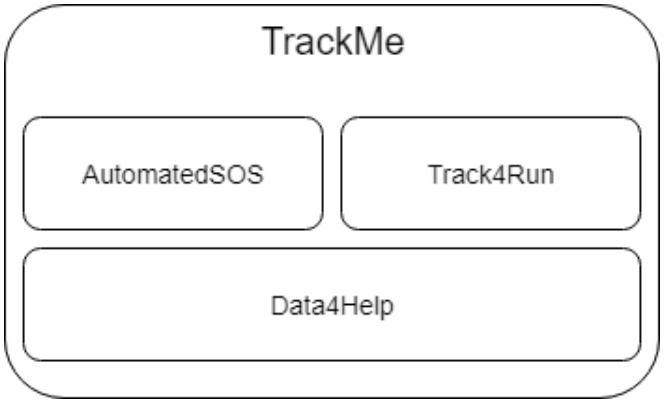


Figure 1: TrackMe ecosystem

Data4Help is the underlying backbone of our system that collects data from end users and makes it available to third party applications.

The other two subsystems are built on top of Data4Help and provide to the user different services: AutomatedSOS offers an automated way of calling an ambulance in case of emergency, while Track4Run gives to the end users a platform to organize and participate to run events.

The following subsections give a more detailed analysis of the domain of each subsystem and are accompanied with the corresponding class diagrams. The aim of these diagrams is to identify the main actors and components of the subsystems and the relationship between them.

2.1.1 Data4Help

Data4Help is meant to be a stand-alone system which works as a gateway between user data producers and third parties, which are data consumers. The central actor of this system is the user.

Each user can configure multiple data sources: each of them can collect different parameters that together form the user data. On the other hand, third parties can make some data requests, that can be targeted to a single user data or a group of data.

These requests can be of two types: subscriptions, which will cause the third party to be updated whenever the chosen data set changes, or single requests.

Bearing these considerations in mind, the following diagram is given to better describe the domain of the application.

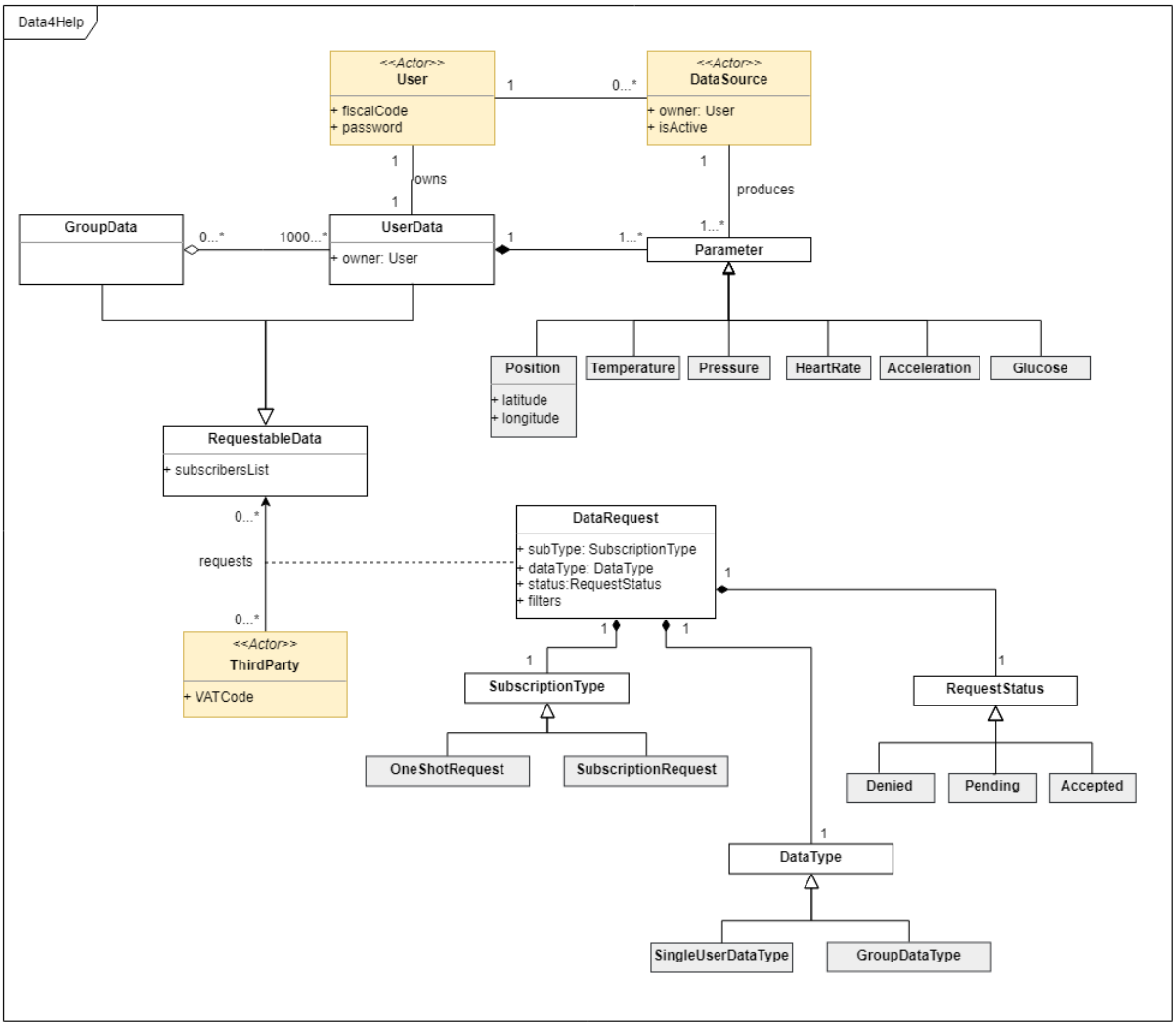


Figure 2: Data4Help Class Diagram

Please note that the parameter types listed here are the most commonly used in many kind of applications, but this doesn't exclude that more parameter types can be added in future phases.

2.1.2 AutomatedSOS

Differently from the previous one, AutomatedSOS is an application which relies on other systems, such as an Ambulance Calling System and Data4Help, to provide a specific service to the customers. The offered service is a monitoring system for Data4Help users, in which some threshold values can be set for each vital parameter. If one of these parameters exceeds his thresholds, this will cause the system to call an ambulance.

In this perspective, the AutomatedSOS system can be seen as one of the third parties of the previous diagram: for each new user, it makes a subscription request for the data of that user to the Data4Help system

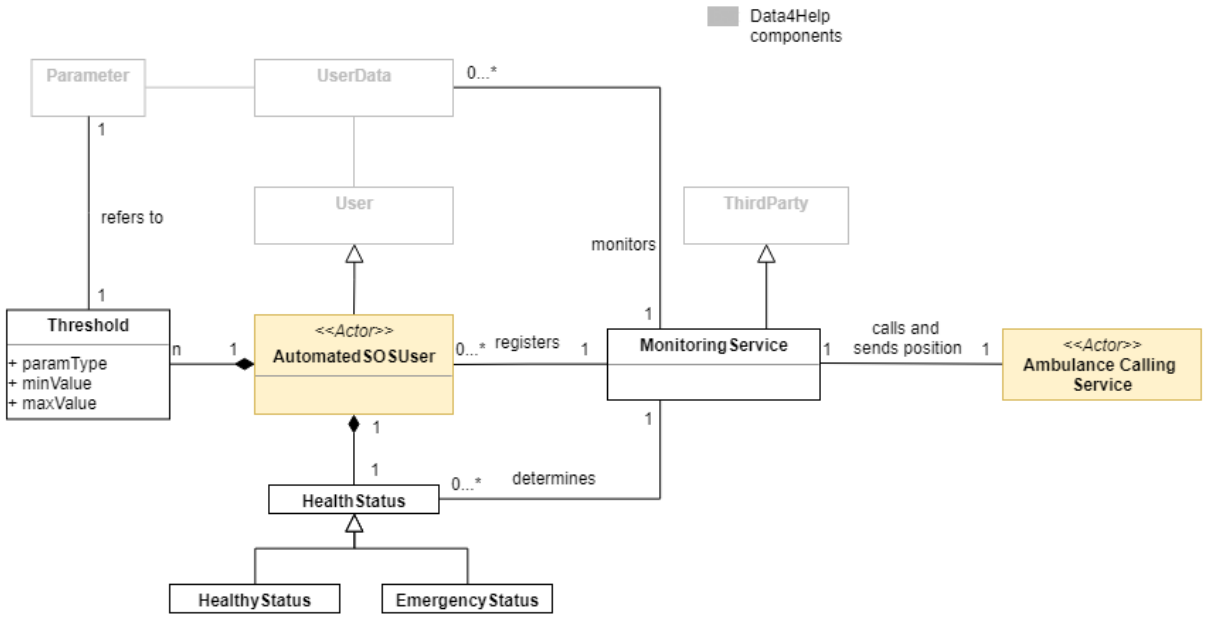


Figure 3: AutomatedSOS Class Diagram

2.1.3 Track4Run

Like AutomatedSOS, Track4Run is an application that uses Data4Help to monitor some of its users, in particular the runners who participate to run event. These run events are created and managed by organizers. On the other hand, spectators can see the position of the runners by using the dedicated service.

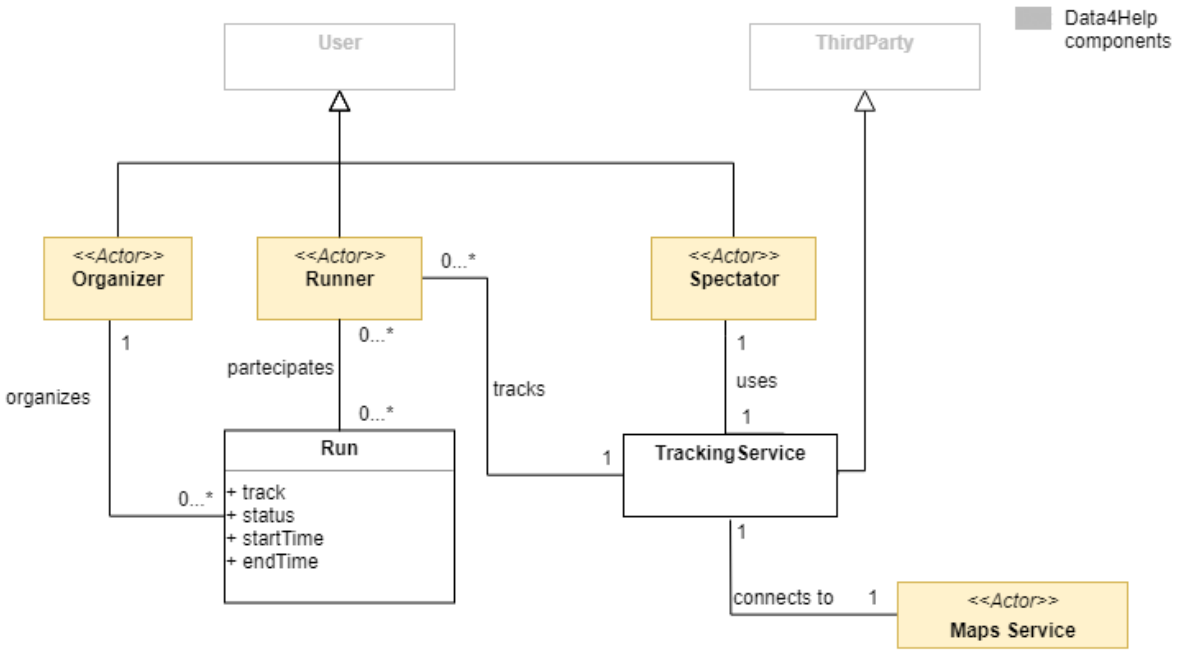


Figure 4: Track4Run Class Diagram

2.2 Product functions

Considering all the goals that the proposed system has to accomplish, an additional and more detailed description is here provided for the main functions that each subsystem has to accomplish.

2.2.1 Data4Help - User Data Acquisition

Data acquisition is the focus of Data4Help and it is also fundamental for all the subsystems built on top of it.

In order to provide this feature, the system must give to the user the possibility to store his position and state of health and share them with the interested third parties.

This kind of data can be acquired from different data sources, such as external applications, smartphones, smartwatches or other similar devices. In order to collect the data from them, the user must firstly synchronize the device or application with Data4Help by providing his credentials and accepting the new data source.

The data source will then be able to retrieve real-time data from the user, and send them to the Data4Help system, which in turn makes them available to the authorized third parties.

In any time the user must be able to add a new data source, decide which parameter can be monitored from which device, add a new data source or stop an existing one from sending new data.

2.2.2 Data4Help - User Data Sharing

The system gives the possibility to third parties to register to the service and request users' data.

Data requests can be done by using either a web interface or an API, which will be provided in order to encourage third parties to develop their own applications that integrate with Data4Help services.

Requested data can be of two different types: single users' data or multiple users' data. The first ones are forwarded to the user and need to be explicitly accepted by him before being carried out. On the other hand, multiple users' data requests do not need user approval, but they can be completed only if the number of users satisfying the search filters are enough to guarantee a proper data anonymization.

In order to be continuously updated, third parties can also make data subscription requests. In this case a callback interface must be provided in order to be notified as soon as the monitored data changes.

Also in this case, the user can at any time decide to revoke to the third parties the permission to access his data.

2.2.3 AutomatedSOS - Health Emergency Monitoring

AutomatedSOS continuously monitors the health parameters selected by the user, exploiting the Data4Help system's data. The user of this system is provided with the possibility to set some thresholds, that can be default or changed at any moment.

As soon as one of them exceeds its pre-defined threshold, the system must contact an external interface to call an ambulance and share with the rescuers the position and vital parameters of the user.

Between the emergency detection and the ambulance calling, the user is notified that a threshold has been exeeded and a small window of time is granted to the user to abort the operation, so to minimize false-positive situations.

2.2.4 Track4Run - Run Events Management

Run events are the central focus of the Track4Run subsystem. Its main goal is to provide the users with the possibility of organizing, participating and watching runs.

Each user will have the possibility to see all the runs that he has created and to create a new one. Creating a run will mean decide a date and hour for starting and ending and select a track from a map: for this, the access to an external service of map visualisation like Google Maps will be necessary. This run can then be later deleted or postponed, or the track can be changed, and on each of these changes the participant runners shall be informed.

The system also offers the possibility to search for ongoing runs an other planned runs that can be joined. In the first case, the user can decide a run to monitor and a map will appear with the live position of all the runners that are involved in that event. In the second case, joining a run means that somebody can track your position while you are running and you will receive notifications if some details change before it starts.

The position tracking is made using Data4Help, so the user's corresponding credentials are needed in order to use this application.

2.3 User characteristics

2.3.1 Actors

In the following section a more specific characterization of the users and actors of each system is provided, based on the domain description that was previously offered.

- **Data4Help User**: in the Data4Help system the user is only a provider of information. After registration, he can access to the platform's functionalities by logging in the system.

- **Third Party**: any organization or software that wants to access the user's data. This is the main direct user of the Data4Help services. He can access the system by registering with a trusted authentication method and using the token generated by his registration. AutomatedSOS and Track4Run are considered to be third parties according to this definition, even if they don't belong to an external organization.
- **Data Source**: any application or device that can be configured by the user to synchronize itself with the system, meaning that it can be authenticated using the user's Data4Help credentials. We can imagine that already existing companies of the wearable and smart devices sector can make a deal with TrackMe to enable the users of their systems to add a Data4Help account, therefore synchronizing the device/application.
- **AutomatedSOS User**: is imagined to be a Data4Help user who wants to carefully monitor his health state, such a an elder user or someone who is undergoing or recovering from an illness.
- **Ambulance Calling System**: a supposedly existing external system that is able to provide an interface to access the national healthcare system, used by AutomatedSOS.
- **Run Organizer**: a user who creates a run event. He must be a Data4Help user who has downloaded the Track4Run application.
- **Runner**: a user who participates to an organized run. Like the previous one, he has to be already registered to Data4Help in order to use the application.
- **Run Spectator**: a Data4Help user that accesses the Track4Run application to follow an ongoing run event.
- **Map Application**: an external service that can provide reliable visualization and tracking service of GPS positions on a map, for example Google Maps. This must be accessed from the Track4Run application through some kind of interface.
- **System Administrator**: someone who is in charge of administrating and mantaining the system. His credentials are generated by the application itself before the deployment, hence he doesn't have to register.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain Assumptions

- **[D1]** Fiscal code uniquely identifies a user of the system.
- **[D2]** Every user owns at least one device capable of retrieving correct real-time health parameters and location
- **[D3]** User devices must grant to the system access to the requested data
- **[D4]** User devices must be continuously connected to the internet
- **[D5]** VAT code uniquely identifies a third party in the system
- **[D6]** Third parties know the monitored user Fiscal Code
- **[D7]** Third parties are able to provide callback interfaces
- **[D8]** AutomatedSOS has access to parameters that can be useful for identifying an emergency status

- **[D9]** There exists an API through which AutomatedSOS can pass to an ambulance information about the identity, the location and the vital parameters of the person in trouble
- **[D10]** Track4Run athlete wears the monitoring device during the whole run
- **[D11]** There is system providing an API to visualize GPS data on a map.

2.4.2 Software dependencies

The considered system relies on external software interfaces to accomplish the previously described functions and make the resulting product more lightweight.

- **Ambulance Calling System**

AutomatedSOS must call an ambulance to rescue the user given his geographical position. For the sake of simplicity, we assume that the national health service offers to the external applications an API that automatically calls an ambulance and sends to it the coordinates to reach the mentioned location.

- **City Maps**

Track4Run users need to use city maps to accomplish different tasks: organizers use them to create events and trace the path of the runs, athletes to check the information about the track in the created events and spectators to display the real-time positions of the athletes on the path during the run. An API of this kind could be Google Maps API, which gives accurate real-time information for mapping, navigation and places.

- **Mobile Operating System**

The two subsystems, AutomatedSOS and Track4Run, are mobile applications that run on some physical smart-devices. For this reason, they need an operating system to interface with such as Android or iOS.

2.4.3 Hardware constraints

- **Servers**

Our system needs to store huge amounts of data coming from single users. For this reason a strong underlying server architecture is needed to develop the back-end part of the system. Possibly the resulting architecture can take inspiration from the classic three-tier web application architecture composed of a web server, an application server and a database server.

- **Sensors and Smart Devices**

Data4Help acquires data from the previously defined data sources, which are devices such as smartwatches, smartphones, wearables and other. Hence, even if they are not directlt part of the system, the reliability and availability of the system's data is strongly connected to the existence of reliable sensors and devices that can collect and send data from them in real-time.

- **GPS**

As concerns the geographical position of the users, Data4Help should give it to the subscribed third parties, if requested. Moreover, AutomatedSOS must know it to send correct info to the ambulances and Track4Run use the user's geographical position to display it on the run path to the spectators during the run. For this reason, in order to retrieve a correct value for the geographical position of the user, this last should have at least one active device with GPS enabled.

- **2G/3G/4G internet connection**

Even if the user is not connected via WiFi or Ethernet cable to the internet, our system (in particular AutomatedSOS and Track4Run) has to retrieve user's data in spite of user's current position. For this reason, the system relies on 3G / 4G technology to grant to the user a non-stop internet access.

3 Specific Requirements

3.1 External Interface Requirements

In the following section we list and describe the interfaces that will be provided by the software to external systems and users in order to access its functionalities. This includes user interface mockups as well as a definition of the APIs that will be realized for third party usage.

3.1.1 User Interface

A subset of the possible user interfaces that our platform offers to the customers is here provided. As always we divide the interfaces according to the system in examination:

- Data4Help was designed to be mainly a web application. The structure of each page should be intuitive and user friendly, so that even the less experienced user in web browsing can easily use the system. The web interface is provided not only for the users, but also to third parties in order to better keep tracks or requests status and visualize data. A parallel API is also offered to third parties, which is described in later sections.
- AutomatedSOS and Track4Run are designed to be mobile applications. The following mockups show examples of the typical situations in which the user could find himself after installing the application and using it on his smart-phone. Hereunder mockups for other smart-devices are not included, but the user interfaces will be very similar to these ones.

An example of UI mockups can be found here below.

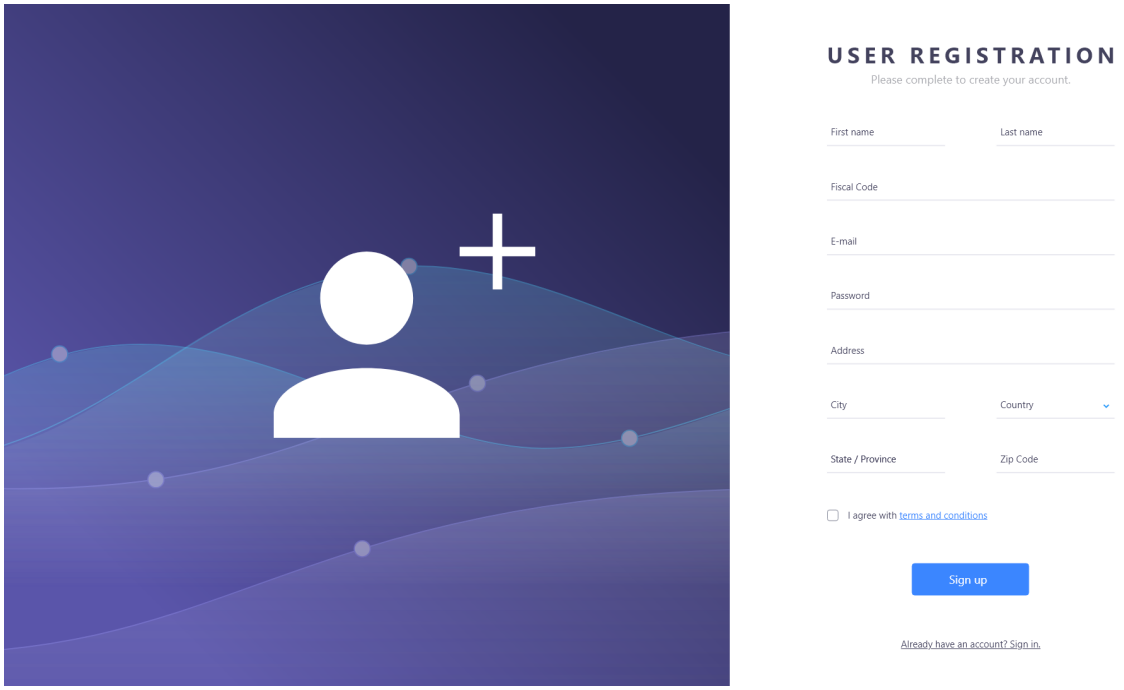


Figure 5: Data4Help - User Registration

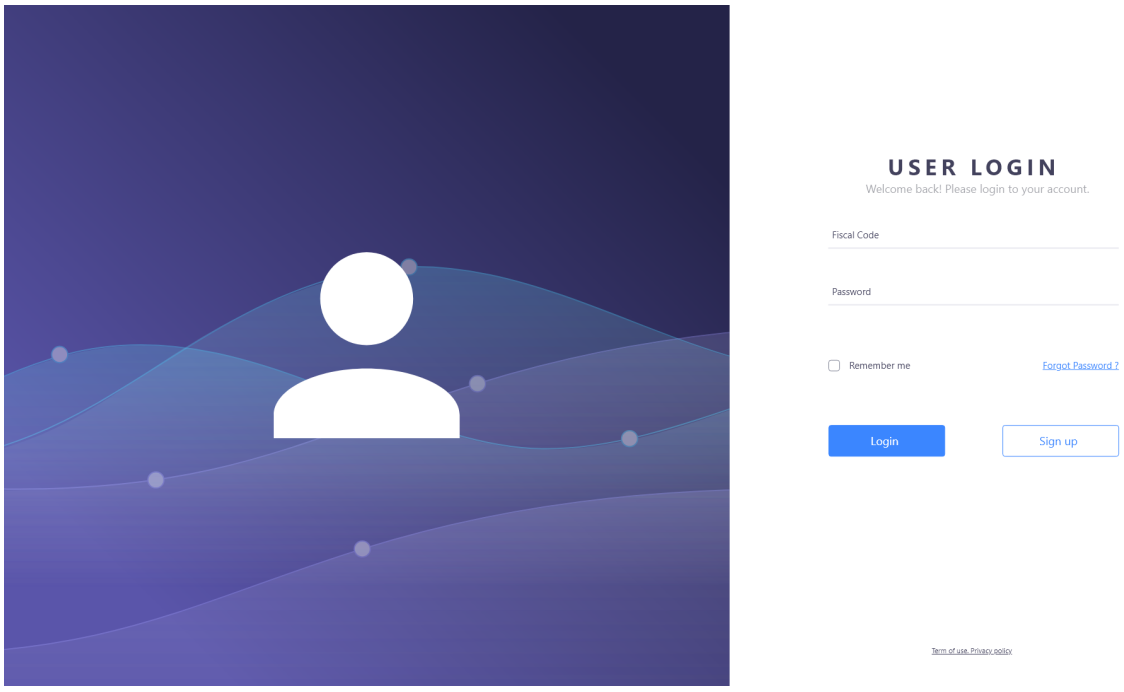


Figure 6: Data4Help - User Login

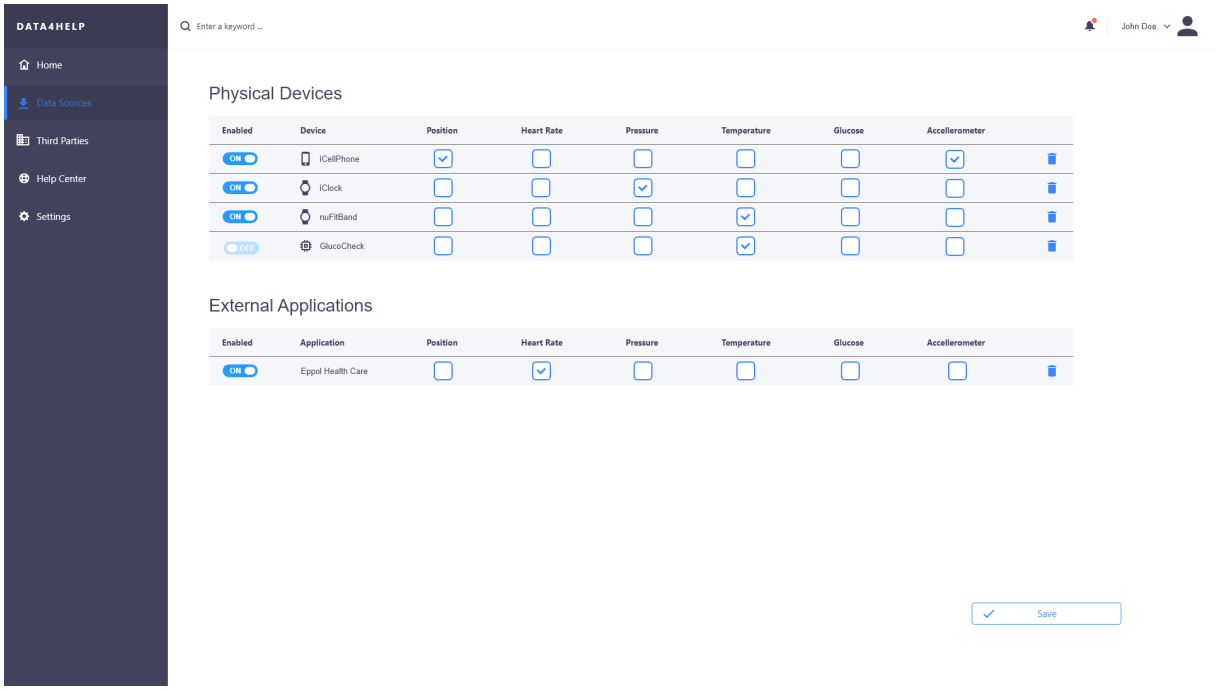


Figure 7: Data4Help - User Data Sources Management

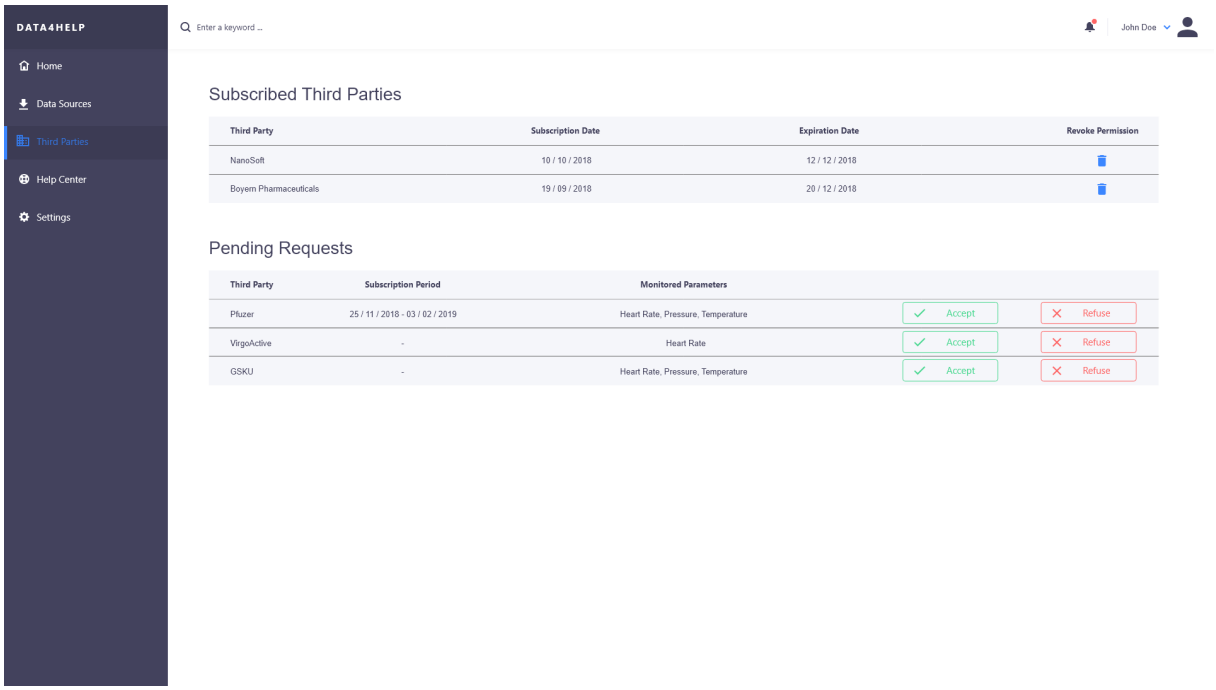
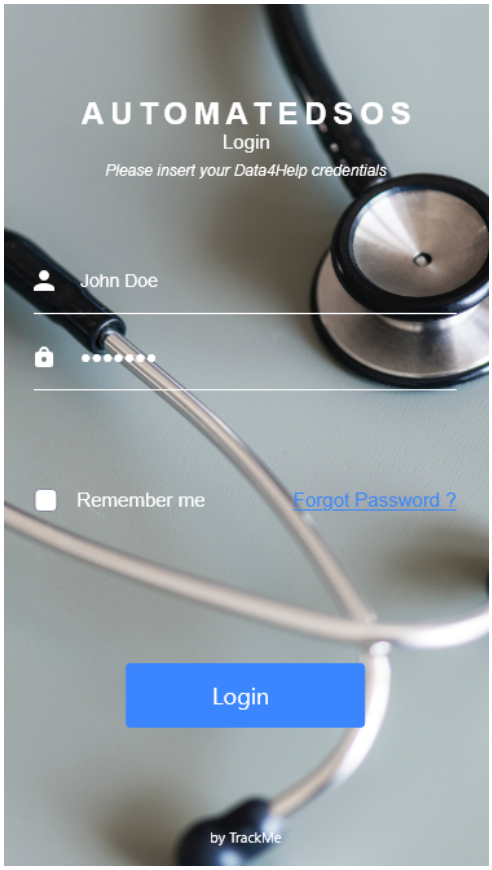
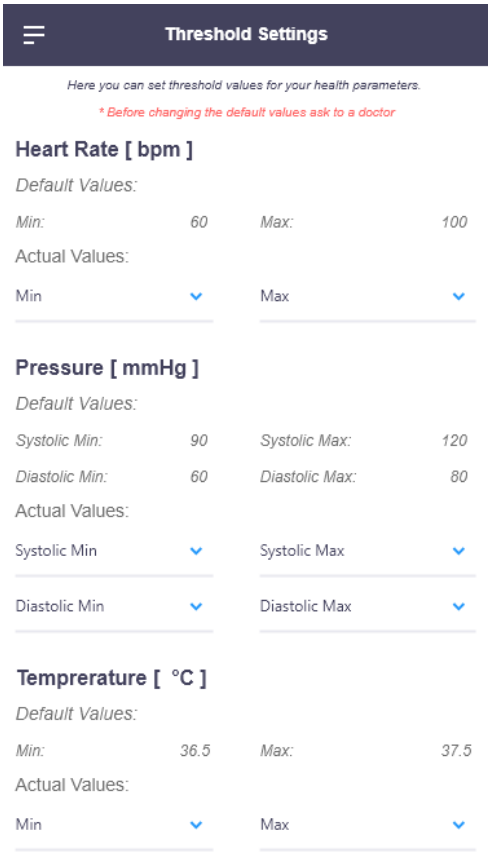


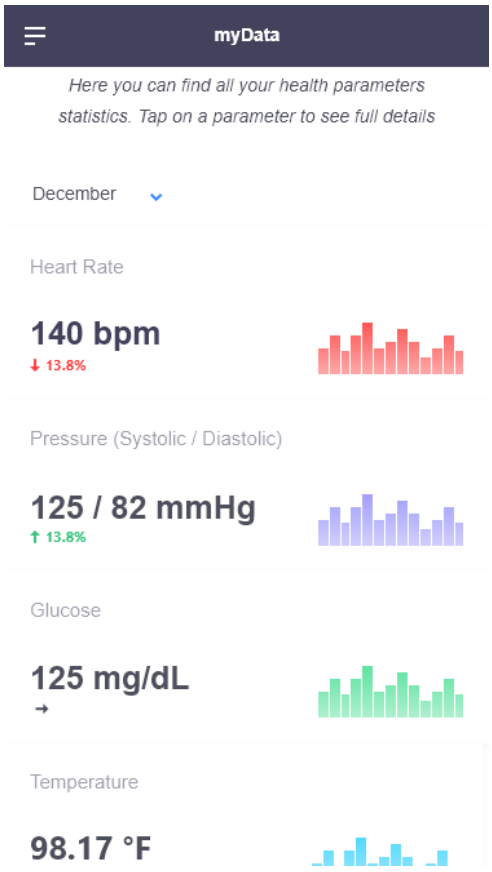
Figure 8: Data4Help - User Third Parties Management



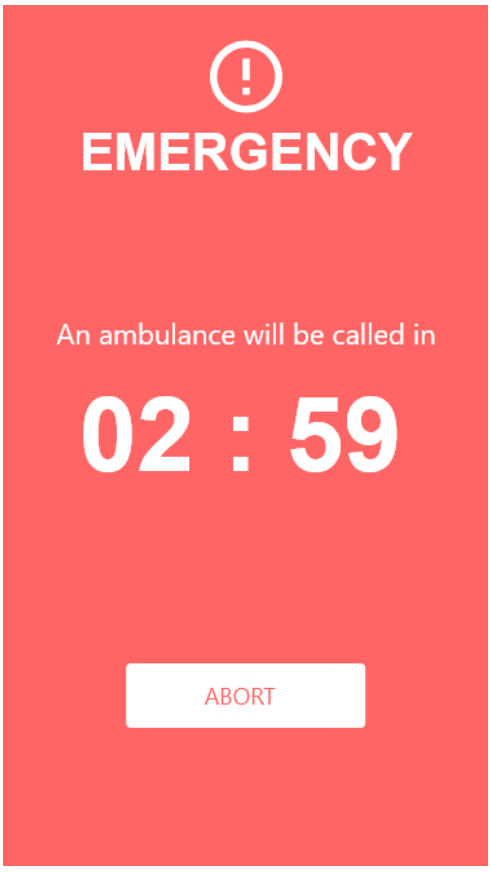
(a) Login



(b) Threshold Settings

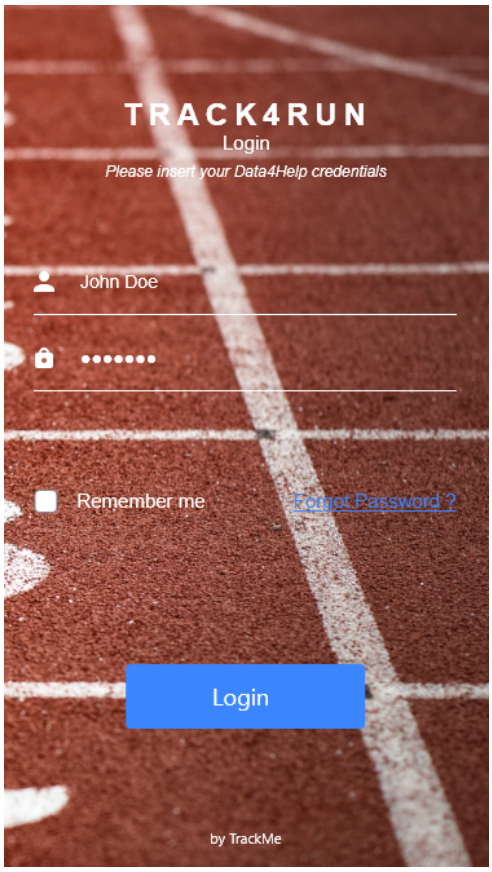


(c) myData

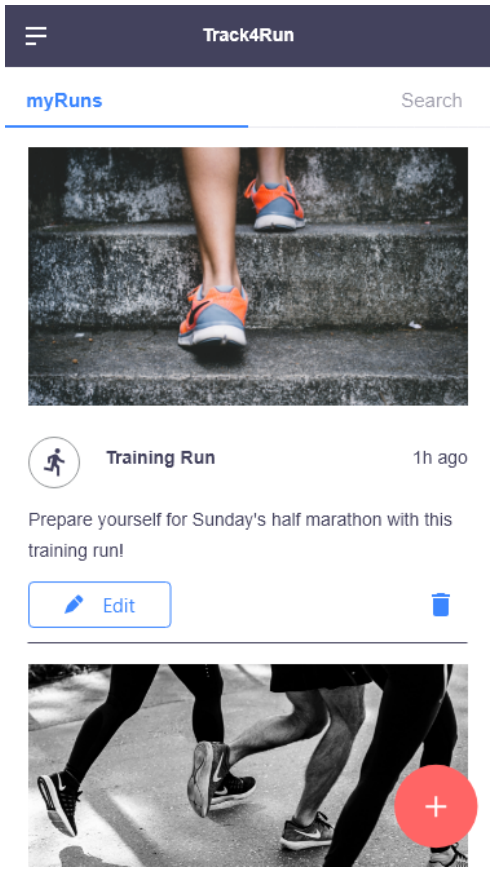


(d) Emergency Alert

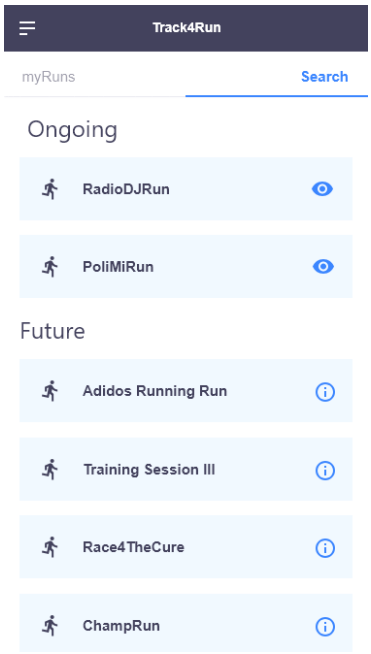
Figure 9: AutomatedSOS Mockups



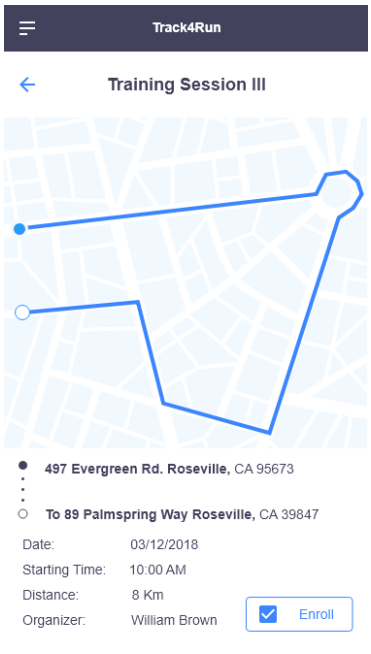
(a) Login



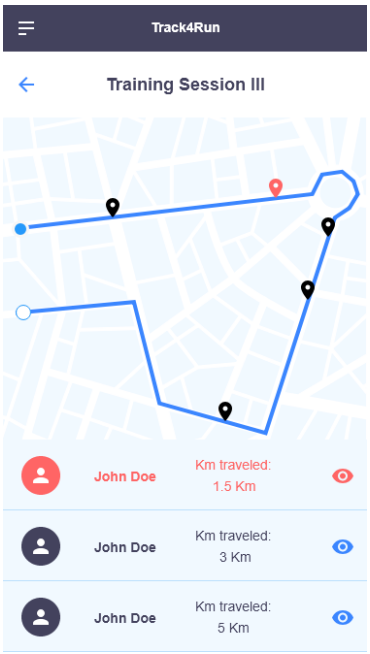
(b) myRuns



(c) Search



(d) Run Details



(e) Live Run Details

Figure 10: Track4Run Mockups

3.1.2 Hardware Interfaces

Since our system is designed to be a software platform running on any compatible hardware, no proprietary hardware interface is provided.

3.1.3 Software Interfaces

The main external software interfaces of our system can be found in Data4Help, which as already explained is the part responsible of connecting data producers to data consumers. To achieve this goal, Data4Help exposes two main software interfaces, which can be thought as APIs that are given to the third party applications in order to connect their software to ours.

- **Data Sharing Interface:** this interface can be used from third party organizations that want to access the user data contained in the system. The interface shall offer four main functionalities: authentication, group data requests, single data requests, subscription requests.

  Authentication must be done using a trustful way of identifying the organization that wants to register to the system, in order to prevent malicious actors to infiltrate the system. This can be for example an electronic signature connected with the VAT code of the company.

  Group data and single user data requests are defined by some filters that the third party wants to apply in order to retrieve only the needed data, and when this is possible they will generate a response containing the latest available data from the system.

  Finally, subscription to new data can be performed by providing a callback interface, that will then be notified with some specific call whenever some new data is available on the system.

- **Data Acquisition Interface:** this interface is provided to the applications and hardware devices that collect data from the user, and gives the possibility to add this kind of information to the user's data set in the system.

  As for third parties, authentication is necessary in this case too. This can be provided by the data source using the user's Data4Help credentials, so that the systems knows which user is adding that data source.

  Once authenticated, the data source is provided with an interface to send data streams to the Data4Help system, that will collect and store them safely.

3.1.4 Communication Interfaces

As communication interfaces, this system will use the commonly available protocols to access the internet and to make remote calls to software interfaces of external and internal systems.

3.2 Functional Requirements

3.2.1 Requirements

[G1] Data4Help must be able to keep track of real time health status and position from registered users' devices

- **[R1]** A User shall be able to register to TrackMe's services providing a Fiscal code and a password of their choice
- **[R2]** A User shall be able to unsubscribe from the service at any moment
- **[D1]** Fiscal code uniquely identifies a user of the system
- **[R3]** After registration, a user must be able to log in by using his/her credentials

- **[R4]** The system must acquire user data only after he/she accepted the data acquisition policy
- **[R5]** The system shall acquire data from external applications that register themselves as data sources of a user
- **[D2]** Every user owns at least one device capable of retrieving correct real-time health parameters and location
- **[D3]** User devices must grant to the system access to the requested data
- **[D4]** User devices must be continuously connected to the internet

[G2] Data4Help should allow third parties to gather information from a single user or from an anonymous group of users

- **[R6]** Third parties must be able to register to Data4Help by providing a VAT code
- **[D5]** VAT code uniquely identifies a third party in the system
- **[R7]** The system shall provide to the registered third parties unique access tokens to use the services
- **[R8]** The system must grant access to a single user data only after his/her confirmation
- **[D6]** Third parties know the monitored user Fiscal Code
- **[R9]** The system must be capable of merging multiple user data and anonymize it
- **[R10]** The system shall grant access to merged data only if the number of individuals whose data satisfy the request is higher than 1000

[G3] Data4Help should allow third parties to subscribe to new data and receive them as soon as they're produced

- **[R11]** The system must store information about third parties' data subscriptions
- **[R12]** The system asks to the third parties who want to subscribe to new data to provide a callback interface
- **[D7]** Third parties are able to provide callback interfaces
- **[R13]** The system must notify every observing third party through their callbacks whenever an observed data changes
- **[R14]** The system must provide an interface to start and manage requests
- **[R15]** The system must give the possibility to third parties to unsubscribe from user data

[G4] AutomatedSOS should be able to identify an health emergency when the user data are below/exceeding a certain threshold

- **[R16]** AutomatedSOS users must be signed in with a Data4Help account
- **[R17]** AutomatedSOS can use data coming from Data4Help to monitor the health condition of its users
- ~~**[R18]** AutomatedSOS, when possible, uses data coming directly from the device sensors.~~
- **[R19]** When activated, AutomatedSOS provides some default values for parameters thresholds

- **[R20]** AutomatedSOS gives the user the possibility to set his/her own threshold
- **[D8]** AutomatedSOS has access to parameters that can be useful for identifying an emergency status
- **[R21]** AutomatedSOS shall be able to unsubcribe from the service at any time

[G5] AutomatedSOS must notify the user and call an ambulance when it detects a health emergency

- **[D9]** There exists an API through which AutomatedSOS can pass to an ambulance information about the identity, the location and the vital parameters of the person in trouble
- **[R22]** The system must notify the user before calling the ambulance
- **[R23]** The system gives the possibility to abort the operation within 5 seconds
- **[R24]** The system automatically calls the ambulance if the operation is not aborted

[G6] Track4Run allows an organizer to create and manage a run.

- **[R25]** Track4Run users must be signed in with a Data4Help account
- **[R26]** Track4Run allows organizers to create a "run event" providing the track, the date and a starting and ending time
- **[R27]** Track4Run allows organizers to modify the details of the created event
- **[R28]** Track4Run allows organizers to delete his created runs
- **[R29]** Track4Run gives the possibility to the organizers to extend editing permissions of an event to other users (co-organizers)
- **[R30]** Track4Run can show to an organizer a list of his past, ongoing and future events

[G7] Track4Run allows a user to partecipate to an organized run

- **[D10]** Track4Run athlete wears the monitoring device during the whole run
- **[R31]** Track4Run gives the user the possibility to look up for future organized runs and join them
- **[R32]** Track4Run gives the user the possibility to unsubscribe from a previously joined run at any moment
- **[R33]** Track4Run must keep track of the position of every participant of the run from the start until the end of the event

[G8] Track4Run allows a spectator to track the position of the partecipants of the run

- **[R34]** Track4Run gives the spectator the possibility to look up for ongoing runs and select the one to follow
- **[R35]** Track4Run shows to the spectator the position of all the participants of the selected run on a map
- **[D11]** There is system providing an API to visualize GPS data on a map.

3.2.2 Scenarios

Scenario 1 While talking, John tells his friend Matt about a new web service called Data4Help that helps you keeping track of your health status. Hence Matt, interested in checking his heart rate, decide to register to it. In order to do this he has to fill a form with his personal information, including his fiscal code, and choose a username and password that will be used as credentials for the login. After filling the form he clicks on the final checkbox to accept the terms and conditions and complete his registration. Thereafter Matt receives a notification saying he has registered successfully to Data4Help.

Scenario 2 Mary, a Data4Help customer, received for her birthday the new Eppol iClock. Knowing that Eppol's products support Data4Help data soure synchronization, she decide to pair her new device with her Data4Help account. She searches for this feature in her iClock and enables it. Soon a confirmation request arrives via email and she confirms it. The device is added to the already available data sources. After that Mary goes on Data4Help webpage, logs in, browse to the "Data Sources" section and sees the new iClock between the available sources. There she assigns the tracking of the "Hearth Rate" parameter to her smartwatch. With the same procedure, she assigns to her smartphone the tracking of the "Position" parameter. She's then satisfied with her preferences and saves them.

Scenario 3 Michael Garcia, Boyer Pharmaceutical CEO, heard about the Data4Help service and in agreement with the Administrative Board, he decided to introduce it in the company. He visits www.data4help.com/ and clicks on the "Business" section in order to register his company as a registered third party of the service. To do this, Michael fills in the registration form with all the requested information about the company, including the VAT Code and the Electronic Signature, and confirms it. A confirmation allert is immediatly show saying the registration procedure has been successfully completed.

Scenario 4 Brian, patient of the Lenox Private Medical Center, was released yesterday. The clinic, registered to the Data4Help service, decides to monitor his health status to see if the treatment he was under reached the desired results. The doctor that was in charge of Brian logs in to the Data4Help reserved page of the clinic. Once logged in he goes under the "Requests" tab. There, with the aid of many checkboxes, he craft a request to monitor Brian's data. As soon as the doctor sends the request, Brian receives an email notification. He then accept the request, clicking the link in the email. The Lenox Private Medical Center is added to the Third Party Management list on Brian's account. A confirmation email, saying that the user accepted the request is sent to the Medical Center. The doctor sees the e-mail and goes in the "Requests" section of the Data4Help personal page of the company, and from there to the accepted ones. After clicking on Brian's answer, all the requested data are displayed on screen.

Scenario 5 Pfuzer, a big pharmaceutical company registered to Data4Help service, needs to gather the heart rate data of all the italian young people under 30 years old for a market analysis aimed at evaluating the production of a new drug against heart disease. For this reason Todd Chavez, the marketing manager of the company, goes to the Pfuzer personal Data4Help page and once logged in he clicks on the "Requests" tab in the home page. Once in the page he fills a form, filtering the location of the search. Then he tries to send the request but a warning message is immediately displayed on screen saying that the request cannot be satisfied. For this reason, Todd decides to untighten the search parameters. After changing them, the constraints are satisfied and the requested data is immediately made available under the Accepted tab.

Scenario 6 Andrew decided to buy the new iClock as a gift for the birthday of his grandfather Leonard, 93 years old, who suffers from severe heart problems and lives alone. Since Leonard would like to be autonomous, his new device with AuotmatedSOS service active on it, grants him a safer life.

One night Leonard wakes up with severe chest pains. The iClock immediately detects the "heart rate" parameter exceeded the maximum threshold and shows a noisy emergency alert, saying that an

ambulance will be called if the user doesn't abort in the following minute. Leonard is really sick and is not able to abort the operation. In few minutes an ambulance arrives and Leonard is immediately rescued.

Scenario 7 William suffers from epilepsy. The attacks are not very frequent but they are so strong that a few months ago he fell and he got a nasty head injury. While looking for an automated solution to check on his conditions, he hears about AutomatedSOS and decides to activate the service provided by Data4Help.

At first he sets the threshold of the tracked parameters with the help of his doctor and he adds the contacts of his parents. When the wristwatch detects repetitive shaking motion, it automatically sends the user's bluetooth-connected phone text and call alerts to the designated recipients. Within seconds, family members receive these alerts, which include the date, time and GPS location of the event.

Scenario 8 Steve, a AutomatedSOS customer with diabete, needs to periodically check the glucose levels observed by his medical device connected with the AutomatedSOS application on his smartphone. For doing this, he starts the application and goes on the "myData" tab. A nice view appears, containing all the information about his monitored health parameters with a lot of colorful diagrams. Steve filters out the displayed content by selecting the desired observation period. He's then presented with all the data gathered for that period.

Scenario 9 Mario is a personal trainer and he is organizing a run for all his athletes. He is thinking about using the Track4Run Service to accomplish this task. So he download the Track4Run application and logs in with his Data4Help credentials. Once the application is installed and running, he goes under the "myRuns" section and taps on the big plus button.

At this point a configuration frame shows up. Luke starts by filling out a form: he adds the event name, the travelled distance and the starting and ending time. Finally, by clicking on a map, he marks all the checkpoints of the run.

After a quick check, Luke clicks on the "Create" button and a confirmation alert appears saying that the event has been successfully published in the news feed.

Scenario 10 Chris, runner and Data4Help customer, is looking for a run to join. By looking at the news feed of Track4Run on his smartphone he sees the event created by Mario. He taps on the event name and the full description of the event shows up. After reading the description and all the details he decides to join the run, hence he clicks on the "Participate" button. A confirmation alert shows up, saying that the event has been added to the attending events.

Scenario 11 Katy, Chris' wife and Data4Help customer, wants to go cheer his husband at the run. The day of the run she launches the Track4Run and searches for his husband run under the "search" tab. Katy selects the desired run and a nice map with a the GPS position of all the athletes opens up.

3.2.3 Use Cases

Name	User registration
Actor	User
Entry conditions	User enters in the Data4Help web page
Events flow	1. Click on the "Sign Up" button 2. Fill the registration form and the account credentials 3. Accept the terms and conditions by clicking on the checkbox 4. Click on the "Sign Up" button 5. The system elaborate the registration and send back a notification
Exit conditions	Registration is successful and the user is informed via notification
Exceptions	1. The user is already registered 2. There is some invalid data in the form 3. The email is already used 4. Terms and conditions haven't been checked All the exceptions take the user back to the registration procedure

Name	User Login
Actor	User
Entry conditions	User already registered and enters Data4Help web page
Events flow	1. Click on the "Log in" button 2. The user enter his/her credentials 3. Click on the "Enter" button 4. The log in was successful and the user is redirected to the home page of the App
Exit conditions	The log in is successful and the user is redirected to home page
Exceptions	1. Credentials aren't valid The exceptions are notified to the user and the Login procedure restart

Name	Add Data Source
Actor	User
Entry conditions	User synchronizes his device or application with Data4Help
Events flow	1. A confirmation email is sent to the user 2. The user reads the email and clicks on the provided confirmation link 3. The data source is added to user account
Exit conditions	The user confirms the synchronization
Exceptions	1. The user didn't request the synchronization and doesn't click on the confirmation After 24 hours the request is made void and a notification email is sent to the user

Name	Data Sources Management
Actor	User
Entry conditions	User enter in the "Data Sources" section of the web site
Events flow	1. The list of data sources connected to the account is shown 2. The user select which source to configure 3. For that source, the list of all the possible parameter that it can track is shown 4. The user turns On/Off the tracking of each parameter 5. Clicks on the "Save" button
Exit conditions	The user have set his/her preference and saves them
Exceptions	1. the parameter is already tracked from a more reliable source (maybe??)

Name	Third Party Registration
Actor	Third Party
Entry conditions	The third party clicks on "Business" on www.trackme.com/data4help
Events flow	1. Clicks on the "Sign Up" button 2. Fills the form with information regarding the company 3. Clicks on the "Sign Up" button
Exit conditions	Registration is successful and the user is informed via notification
Exceptions	1. Company already registered 2. Electronic Signature not valid All the exceptions are notified and the procedure goes back to registration

Name	Third Party Login
Actor	Third Party
Entry conditions	The third party goes on www.trackme.com/data4help and click "Login"
Events flow	1. Click on the "Login" button 2. Enter the company credentials 3. Click on the "Login" button
Exit conditions	Login is successful and the client is redirected to its reserved page
Exceptions	1. Credentials are not valid The exceptions are notified to the client and the Login procedure restart

Name	One Shot Single Data Request
Actor	Third Party, User
Entry conditions	A third party client is logged in and goes in the "Requests" section
Events flow	1. Clicks on "New Request" button 2. Selects "Single Request" radio button 3. Selects the "One Shot" radio button 4. Inserts the fiscal code of the recipient and the request name in text fields 5. Inserts a brief description of the request by filling a text area 6. Checks which parameters to monitor from a checklist 7. Clicks on the "Send" button 8. The request is notified to the user via email 9. The user checks his email and clicks on the confirmation link 10. The result is notified via e-mail to the third party 11. If the user accepted, the information are made available under the "Accepted" section of the "Request" Data4Help personal page of the company
Exit conditions	The user has responded to the request and, if successful, the data are made available to the third company
Exceptions	1. No user found that correspond to the search 2. The user refuses the request This exception is notified to the third party and the request ends

Name	Subscription Single Data Request
Actor	Third Party, User
Entry conditions	A third party client is logged in and goes in the "Requests" section
Events flow	1. Clicks on "New Request" button 2. Selects "Single Request" radio button 3. Selects the "Subscription Period" from a dropdown 4. Inserts the fiscal code of the recipient and the request name in text fields 5. Inserts a brief description of the request by filling a text area 6. Checks which parameters to monitor from a checklist 7. Clicks on the "Send" button 8. The request is notified to the user via email 9. The user checks his email and clicks on the confirmation link 10. The result is notified via e-mail to the third party 11. If the user accepted, the information are made available under the "Accepted" section of the "Request" Data4Help personal page of the company 12. Every time that the requested data changes in the Data4Help database, the data available to the third party are updated
Exit conditions	The Subscription Period ends
Exceptions	1. No user found that correspond to the search 2. The user refuses the request This exception is notified to the third party and the request ends

Name	One Shot Group Data Request
Actor	Third Party
Entry conditions	A third party client is logged in and goes in the "Requests" section
Events flow	1. Clicks on "New Request" button 2. Selects "Group Request" radio button 3. Selects the "One Shot" radio button 4. Inserts the request name in a text field 5. Specificy the search area by filling a short form 6. Inserts a brief description of the request by filling a text area 7. Checks which parameters to monitor from a checklist 8. Clicks on the "Send" button 9. The result is notified via e-mail to the third party 10. If successfull, the information are made available under the "Accepted" section of the "Request" Data4Help personal page of the company
Exit conditions	The request was successful and the data are made available to the third company
Exceptions	1. Request is rejected due to lack of anonymization (less than 1000 users) The exception notifies the third party on reasons of the rejection and returns to the Data Request page

Name	Subscription Group Data Request
Actor	Third Party
Entry conditions	A third party client is logged in and goes in the "Requests" section
Events flow	1. Clicks on "New Request" button 2. Selects "Group Request" radio button 3. Selects the "Subscription Period" from a dropdown 4. Inserts the request name in a text field 5. Specify the search area by filling a short form 6. Inserts a brief description of the request by filling a text area 7. Checks which parameters to monitor from a checklist 8. Clicks on the "Send" button 9. The result is notified via e-mail to the third party 10. If successful, the information are made available under the "Accepted" section of the "Request" Data4Help personal page of the company 11. Every time that the requested data changes in the Data4Help database, the data available to the third party are updated
Exit conditions	The Subscription Period ends
Exceptions	1. Request is rejected due to lack of anonymization (less than 1000 users) The exception notifies the third party on reasons of the rejection and returns to the Data Request page

Name	AutomatedSOS Login
Actor	User
Entry conditions	The user has previously installed the AutomatedSOS app on his smartphone
Events flow	1. Clicks on the "Login" button 2. Enters his account credentials 3. Click on the "Login" button
Exit conditions	Login is successful
Exceptions	1. Credentials are not valid The exceptions are notified to the client and the Login procedure restart

Name	Monitoring Service
Actor	AutomatedSOS, User, Ambulance
Entry conditions	The user previously installed AutomatedSOS application and logged with his Data4Help account
Events flow	1. The Monitoring Service continuously checks the available real time data 2. Checks for threshold constraints 3. Whenever a parameter is found below or above its thresholds the user is given an emergency status 4. On the user's smartdevice, an emergency status screen with a countdown to call an ambulance shows up 5. An API call for an ambulance in the users' with emergency status location is made
Exit conditions	User unsubscribe from Data4Help or unistall AutomatedSOS application
Exceptions	• Lost physical device for data acquisition This exception notifies the client and pause the Monitoring System until the problem is fixed • User clicks on the "Abort" in the emergency status screen This exception revokes the emergency status of the user

Name	Threshold Settings
Actor	User
Entry conditions	User enter in the "Threshold Settings" tab of the AutomatedSOS application
Events flow	1. The system shows tracked parameters, default values and saved values for each of them 2. For every parametery the user can change the actual value of the thresholds by chosing from a dropdown 3. The user saves the changes
Exit conditions	The user saves and exits
Exceptions	

Name	MyData
Actor	User
Entry conditions	User enter in the "MyData" tab of the AutomatedSOS application
Events flow	1. The system shows all the user gathered info 2. The user can select from a dropdown the observation period 3. Whenever a filer is changed the app respond with the filtered information
Exit conditions	The user exits the "myData" tab
Exceptions	1. the system haven't gathered any info yet The exceptions are notified to the user and the "myData" page is shown with the available data

Name	Create a Run
Actor	User (Organizer)
Entry conditions	The organizer has previously installed the Track4Run application
Events flow	1. The organizer clicks on the "+" icon on the bottom of the "myRuns" section 2. Enter information about the run 3. Select the run path from the map 4. Click on the "Create" button
Exit conditions	The organizer has successfully created a run
Exceptions	

Name	Run enroll
Actor	User (Runner)
Entry conditions	The runner has previously installed the Track4Run application
Events flow	1. The runner clicks on the "Search" tab 2. Choose from the "Future" list of runs which one to enroll 3. Clicks on the "Enroll" button
Exit conditions	The user has successfully enrolled to the run
Exceptions	1. The user doesn't possess a device able to Track the runner 2. The user is in a bad shape for a run, medical advice is suggested This exceptions is notified to the client and the procedure goes on

Name	Run unsubscription
Actor	User (Runner)
Entry conditions	User already enrolled on a Run and he/she want to unsubscribe
Events flow	1. Looks up under "My Runs" tab for the previously joined run 2. Clicks on the "Unsubscribe" button
Exit conditions	User has successfully unsubscribed from the run
Exceptions	

Name	Watch Run
Actor	User (Spectator), Users (Runners)
Entry conditions	A spectator goes under the "Search" section of Track4Run
Events flow	1. Select a Run form the list of ongoing runs 2. Watch the map of the run with the real time GPS tracking of the runners
Exit conditions	Run ends or spectator exits from the application
Exceptions	1. Runner has connection problems 2. Spectator has connection problems 3. Server has connection problems All exception are notified to the spectator and the process goes on

Name	Run modification
Actor	User (Organizer)
Entry conditions	The organizer goes under the "myRuns" tab of the Track4Run application
Events flow	1. Select the run you want to modify from the one presented in the timeline 2. Clicks on the "Edit" button 3. Modify the information of the run 4. Click on the "Confirm" button
Exit conditions	The organizer successfully modified the run and the participants are notified
Exceptions	

Name	Run deletion
Actor	User (Organizer)
Entry conditions	The organizer goes under the "myRuns" tab of the Track4Run application
Events flow	1. Select the run you want to modify from the one presented in the timeline 2. Clicks on the trash bin icon
Exit conditions	The organizer successfully deleted the run and the participants are notified
Exceptions	

3.2.4 Sequence Diagrams

Data4Help Sequence Diagrams

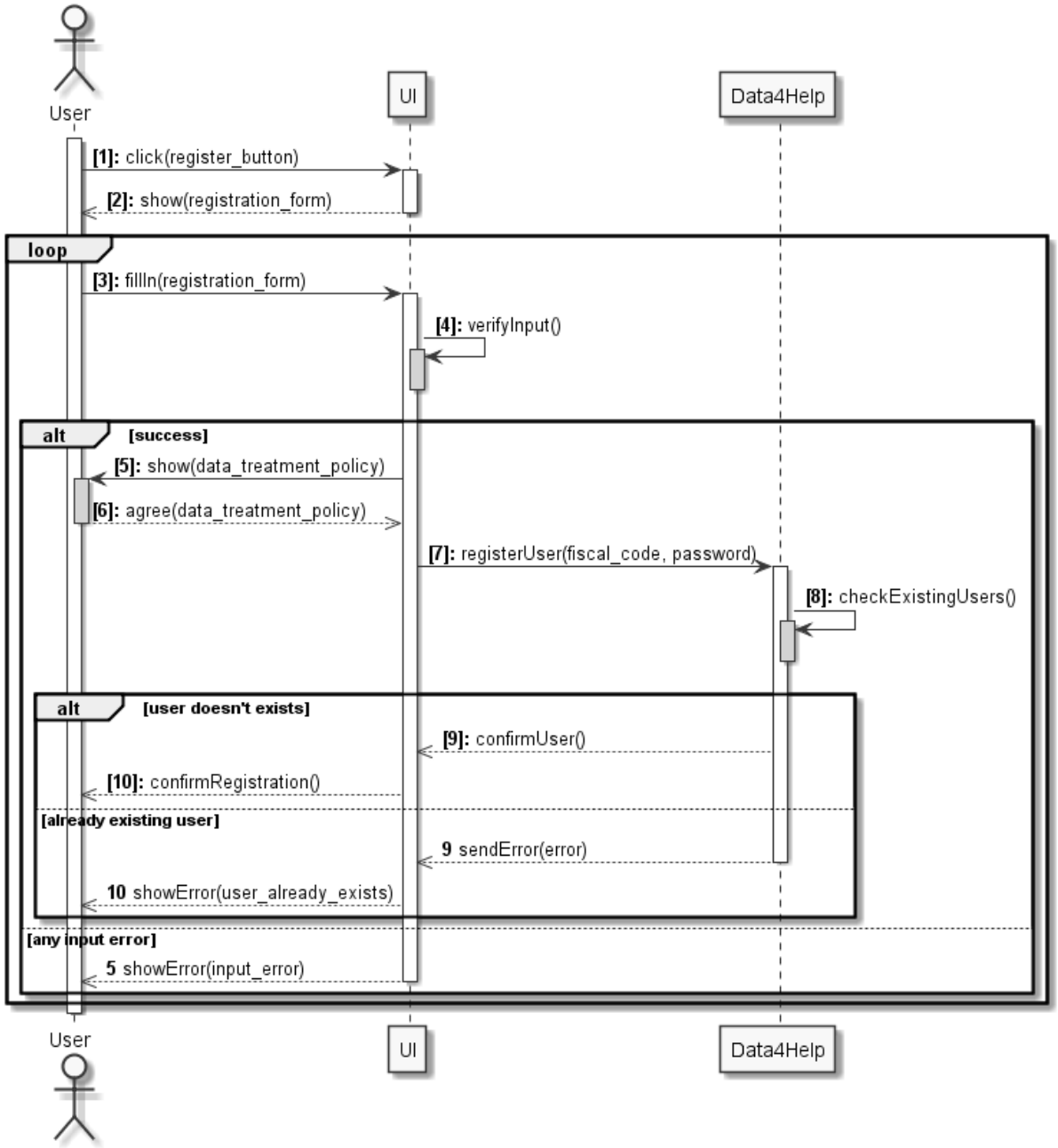


Figure 11: Data4Help - User Registration Sequence

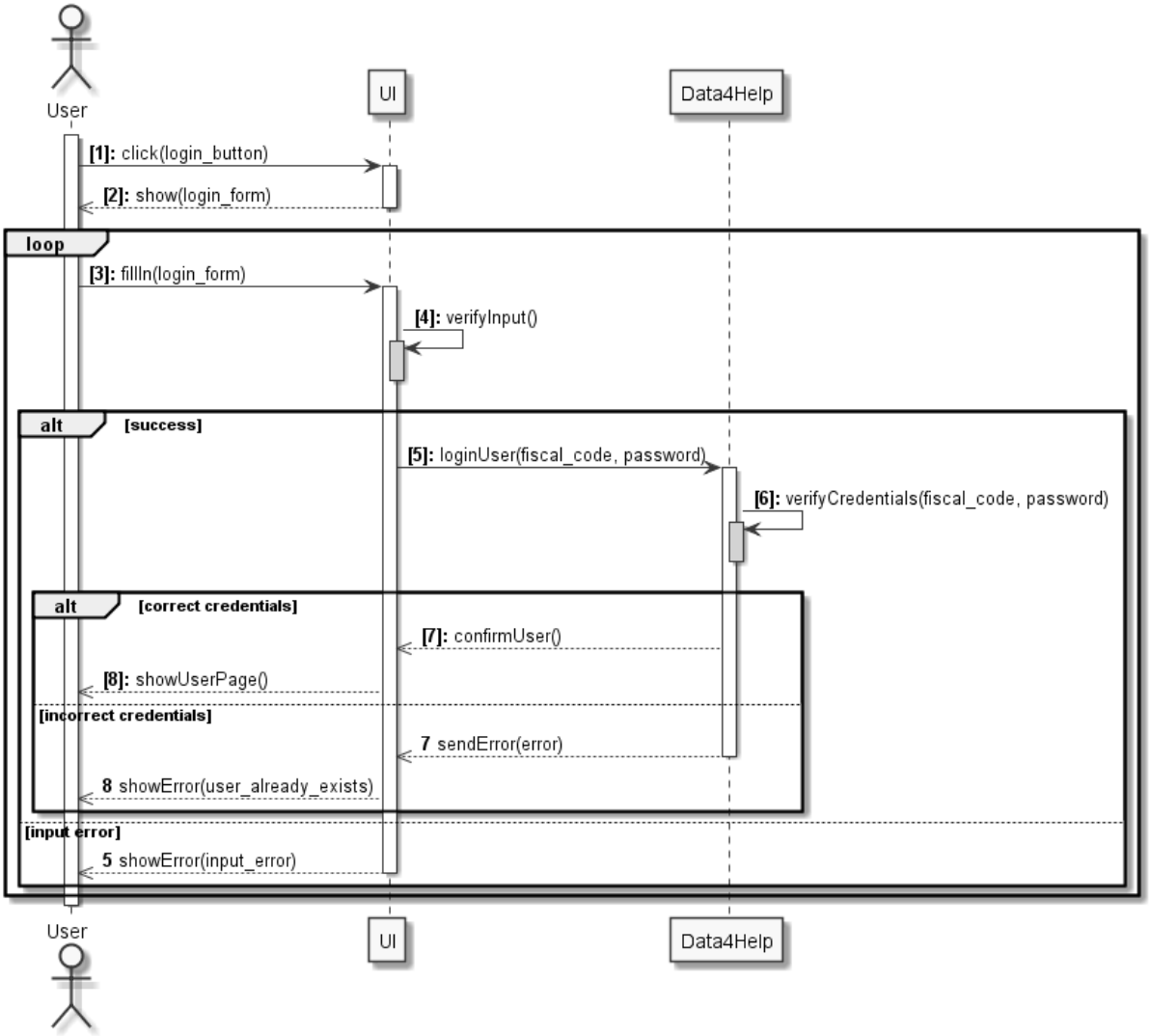


Figure 12: Data4Help - User Login Sequence

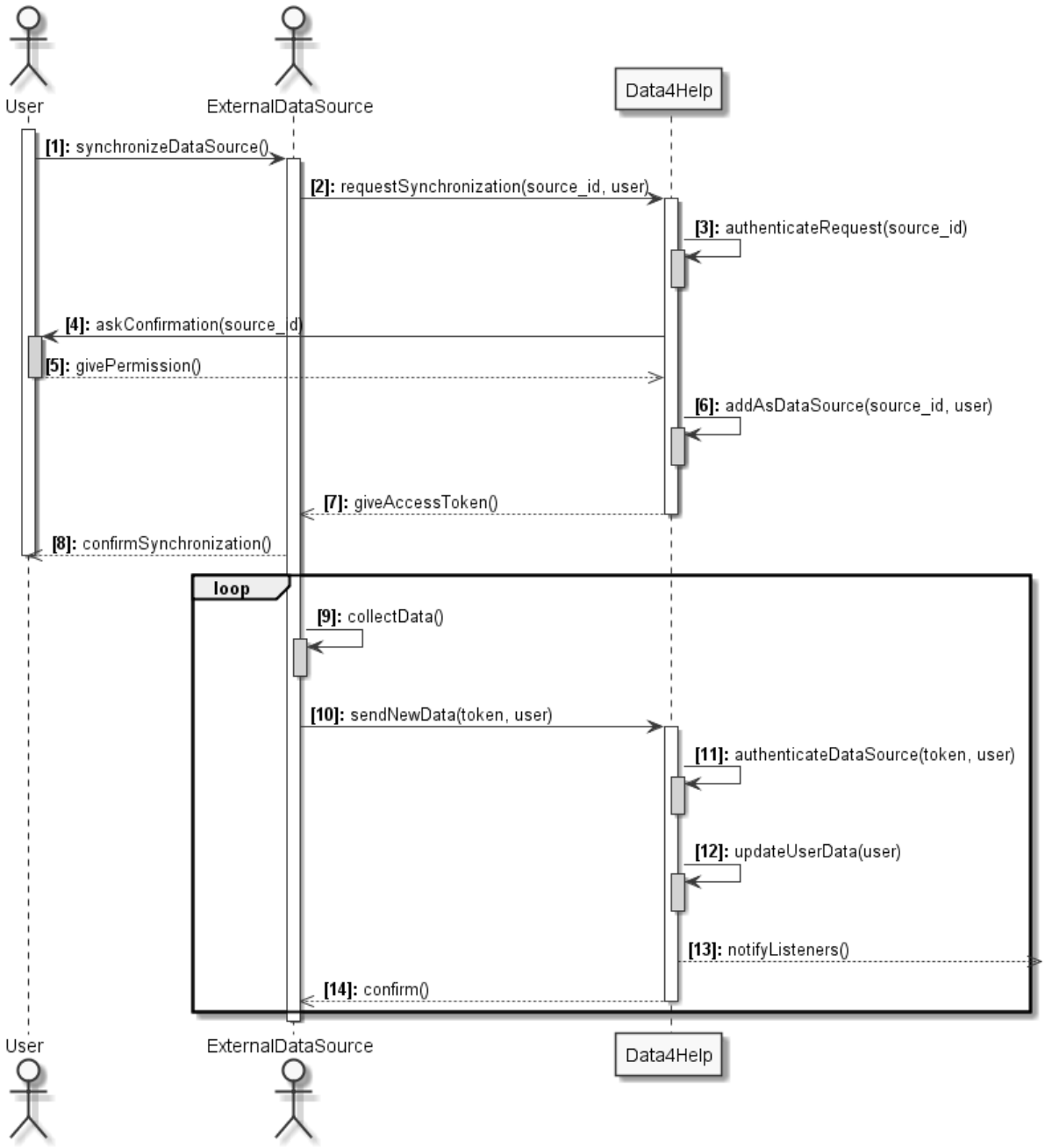


Figure 13: Data4Help - Data Source Synchronization Sequence

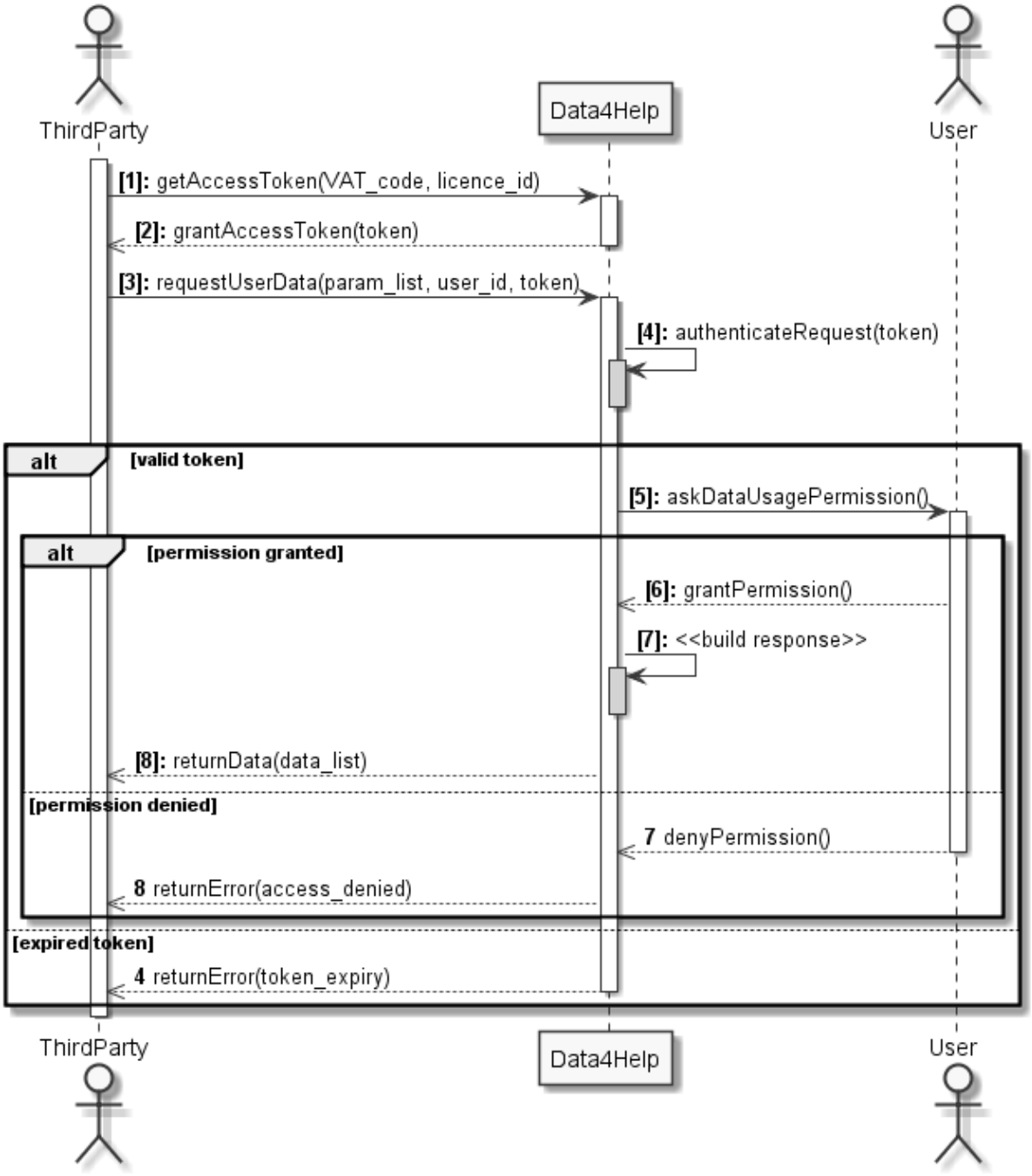


Figure 14: Data4Help - Third Party single data request sequence

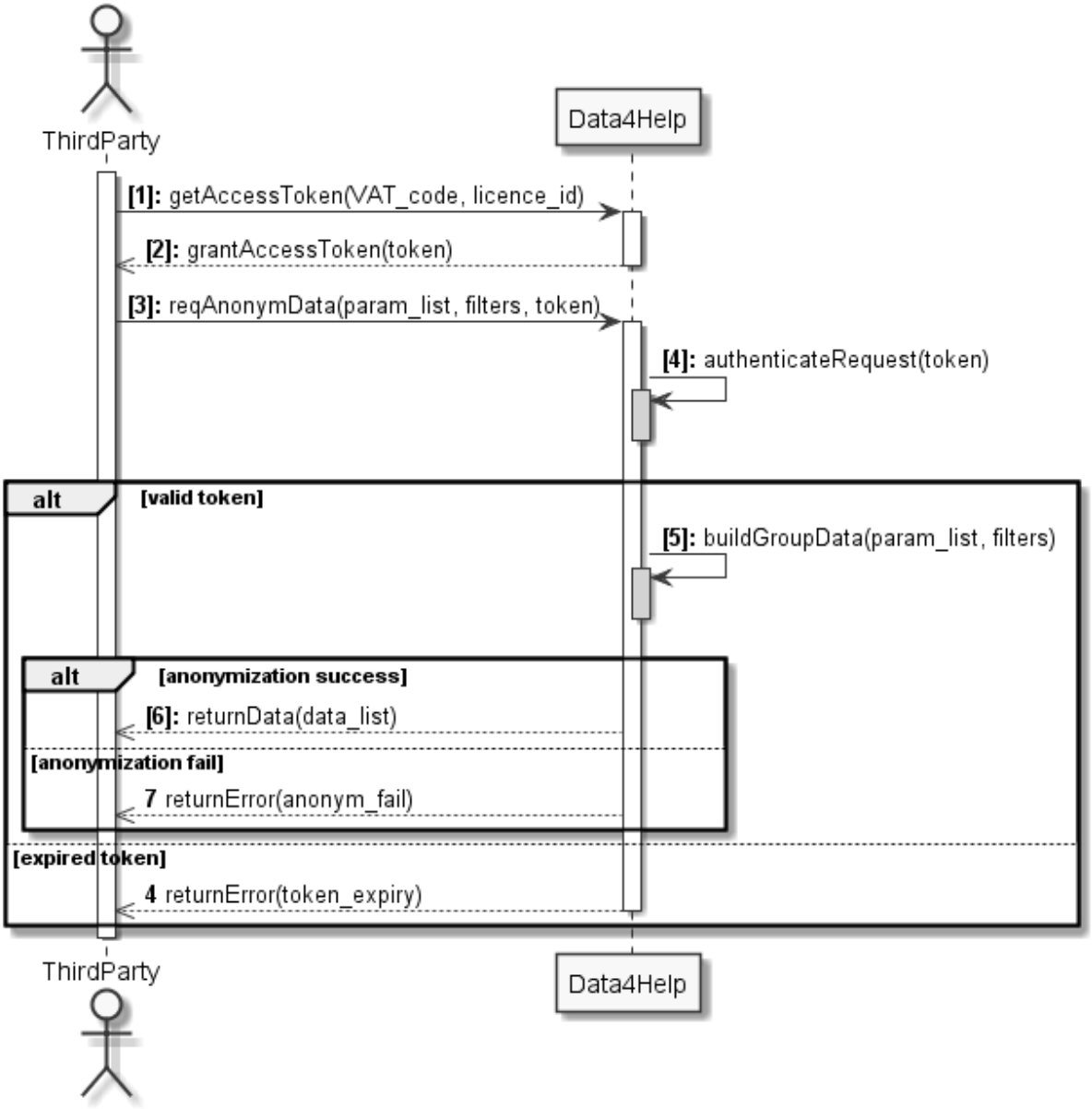


Figure 15: Data4Help - Third Party group data request sequence

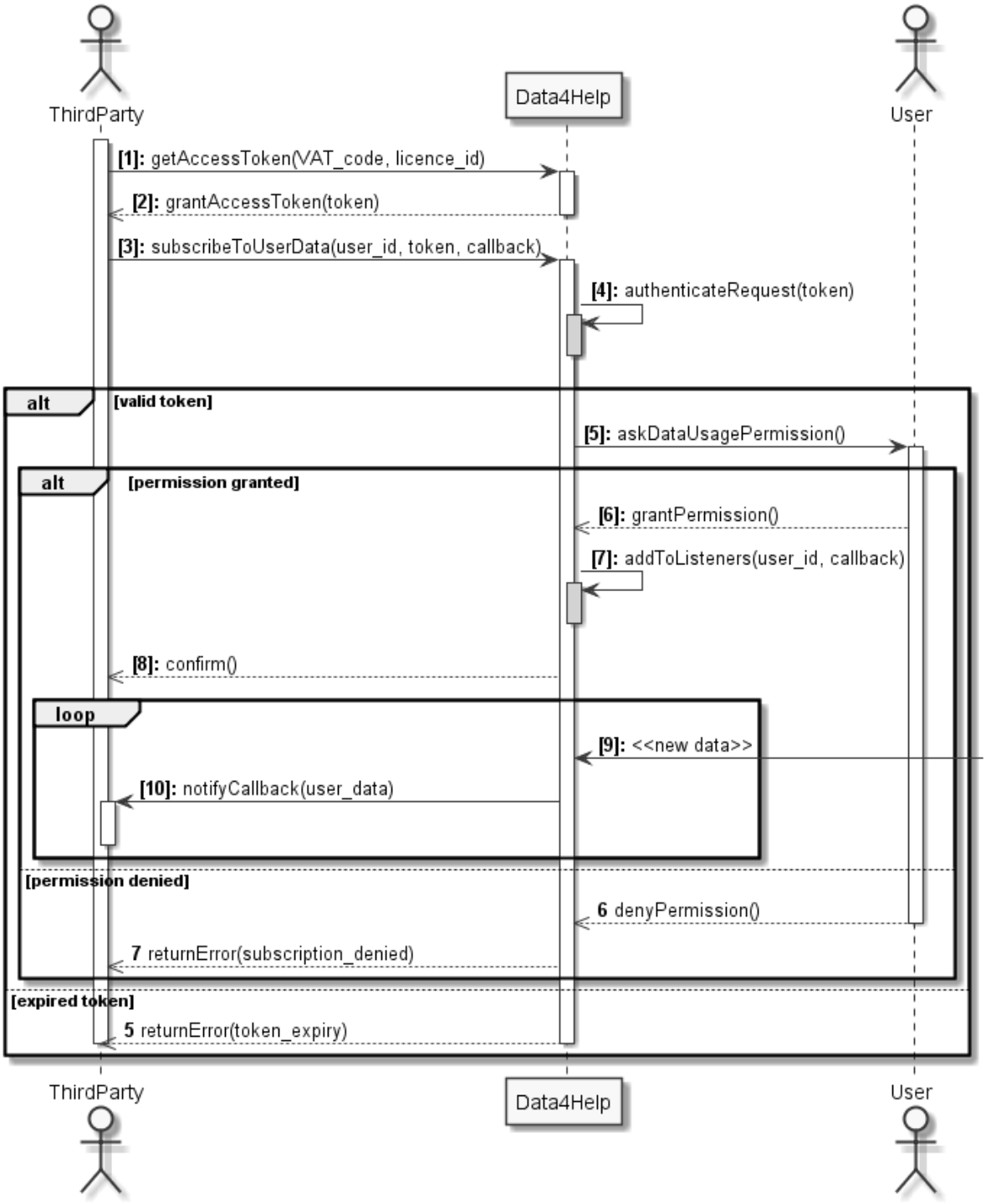


Figure 16: Data4Help - Third Party subscription request sequence

AutomatedSOS Sequence Diagrams

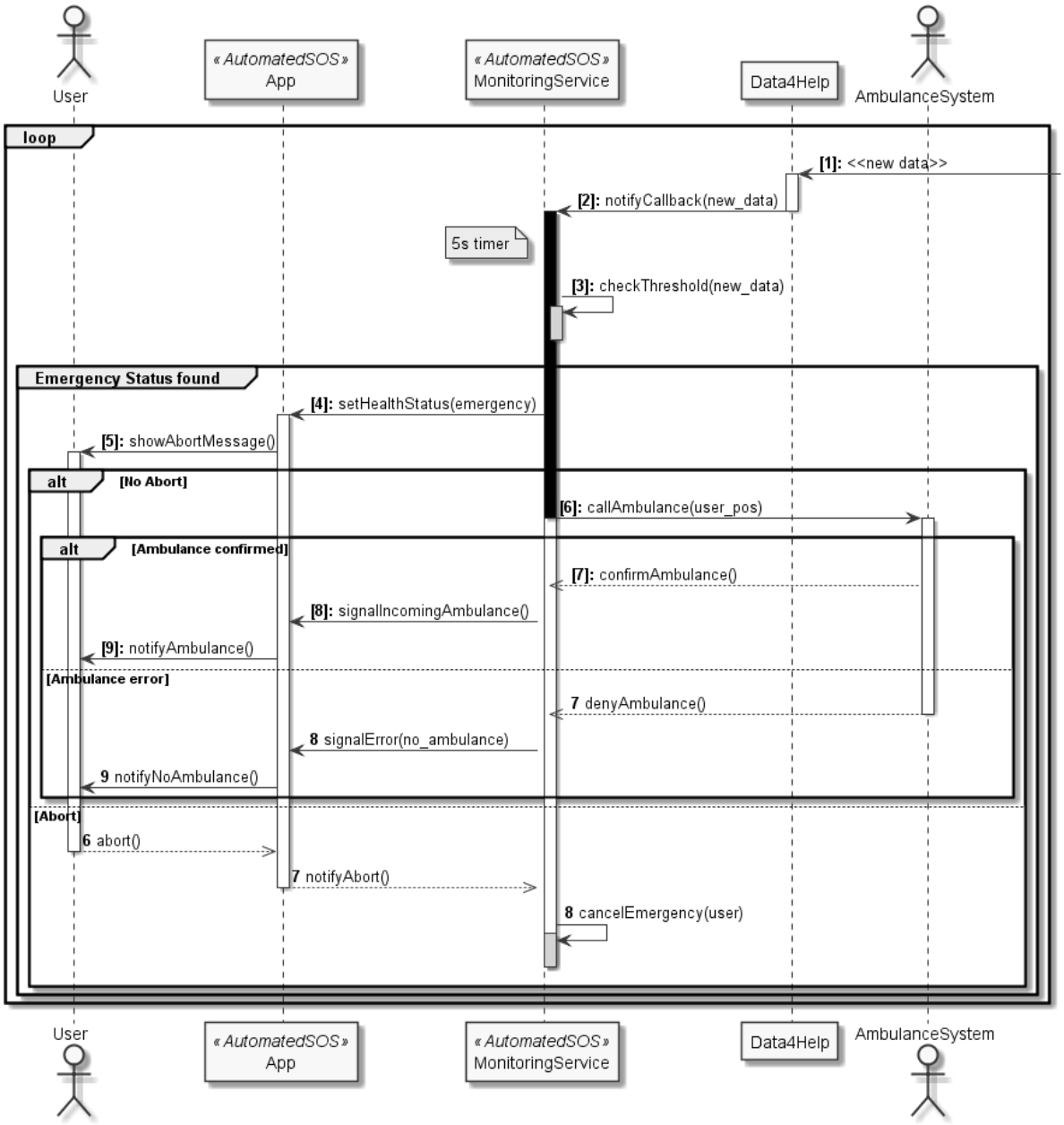


Figure 17: AutomatedSOS - Emergency sequence

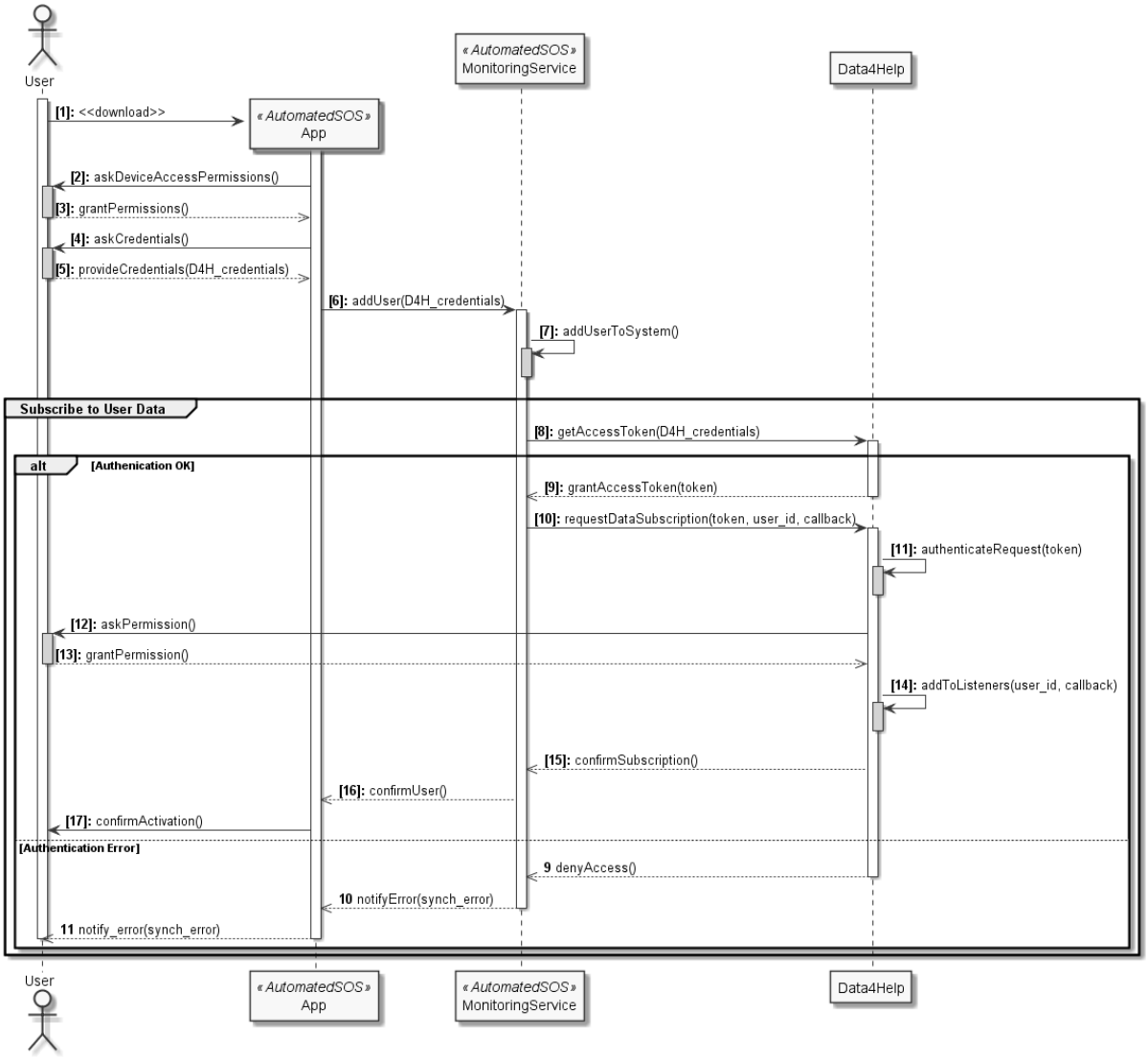


Figure 18: AutomatedSOS - User Registration sequence

Track4Run Sequence Diagrams

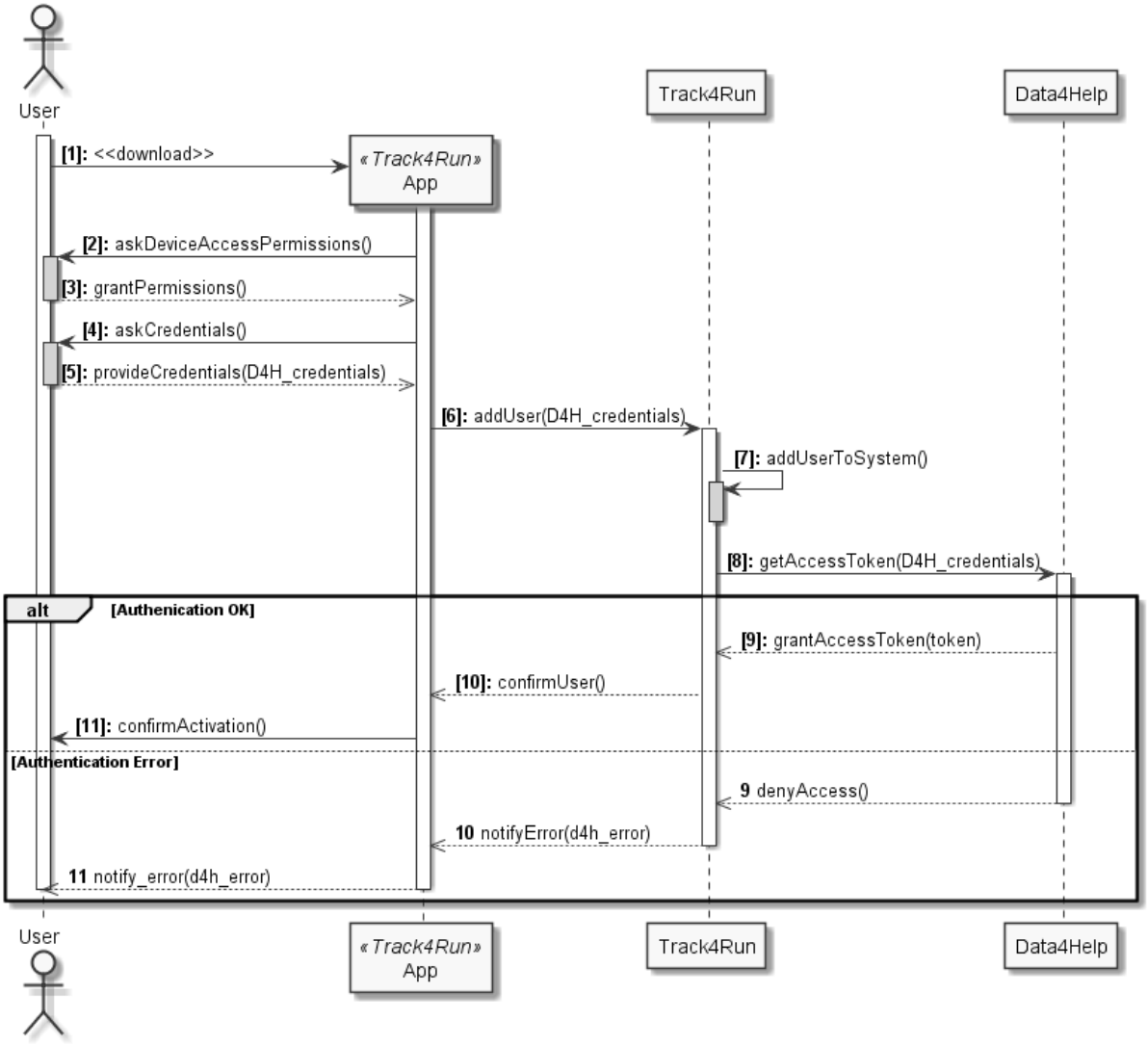


Figure 19: Track4Run - User Registration sequence

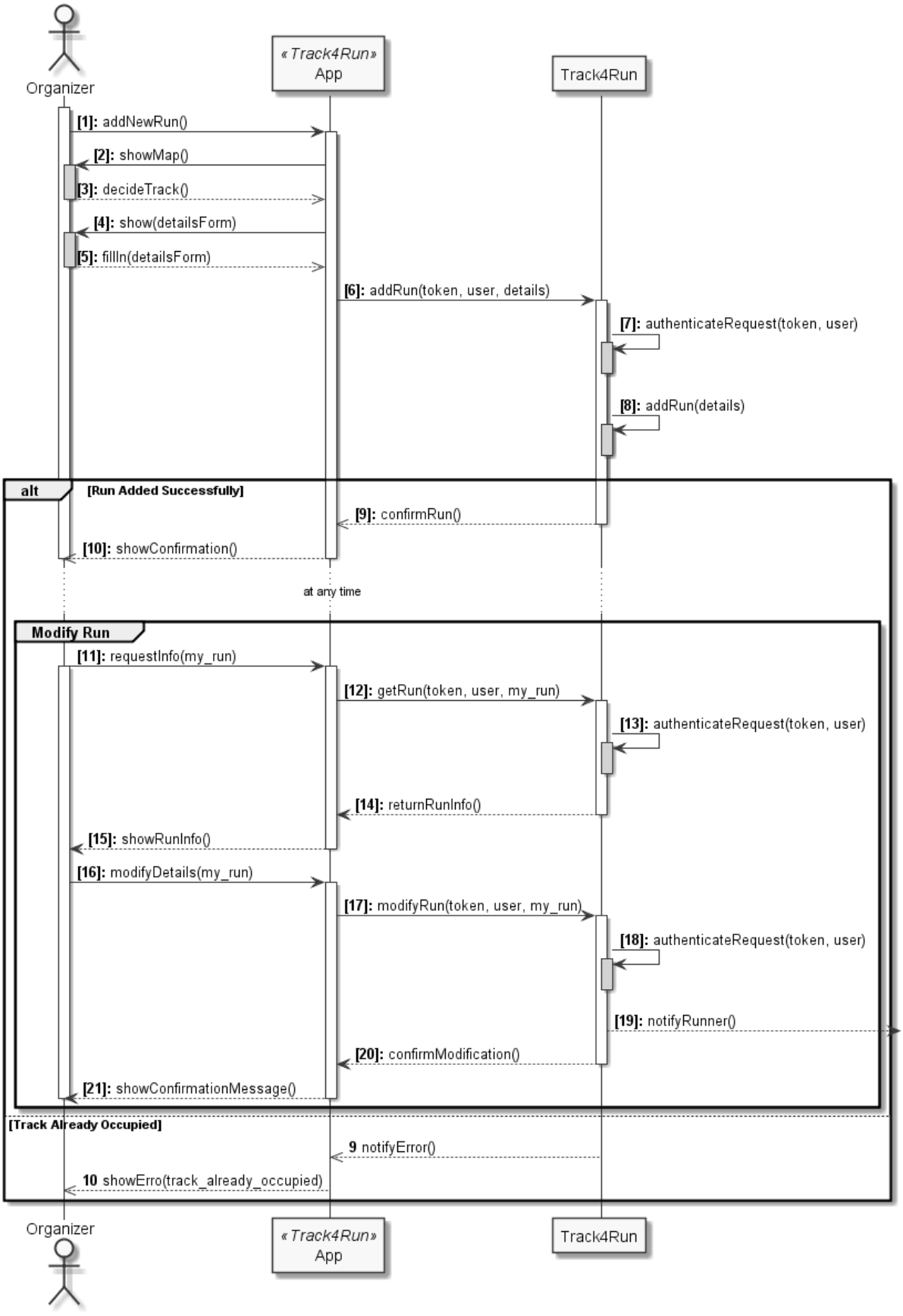


Figure 20: Track4Run - Run Creation sequence

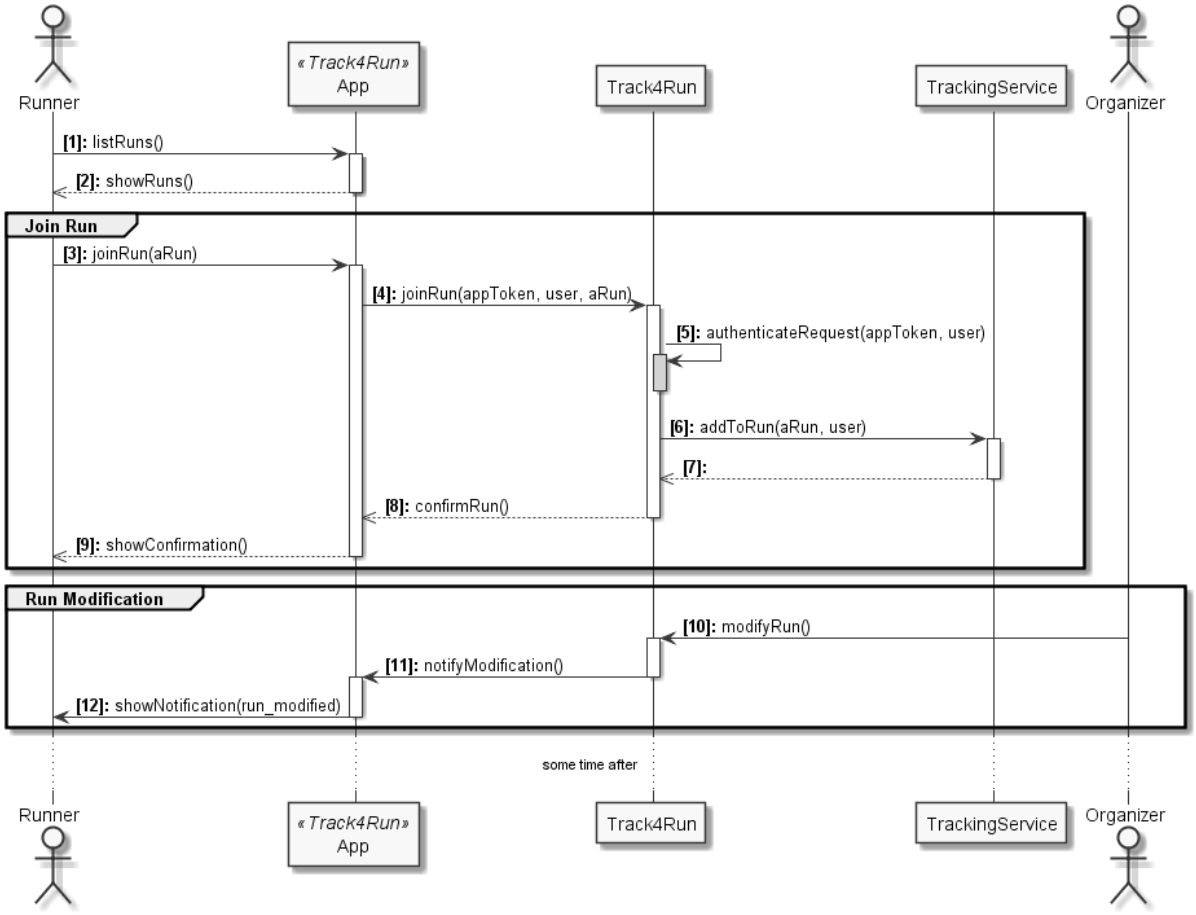


Figure 21: Track4Run - Run joining sequence

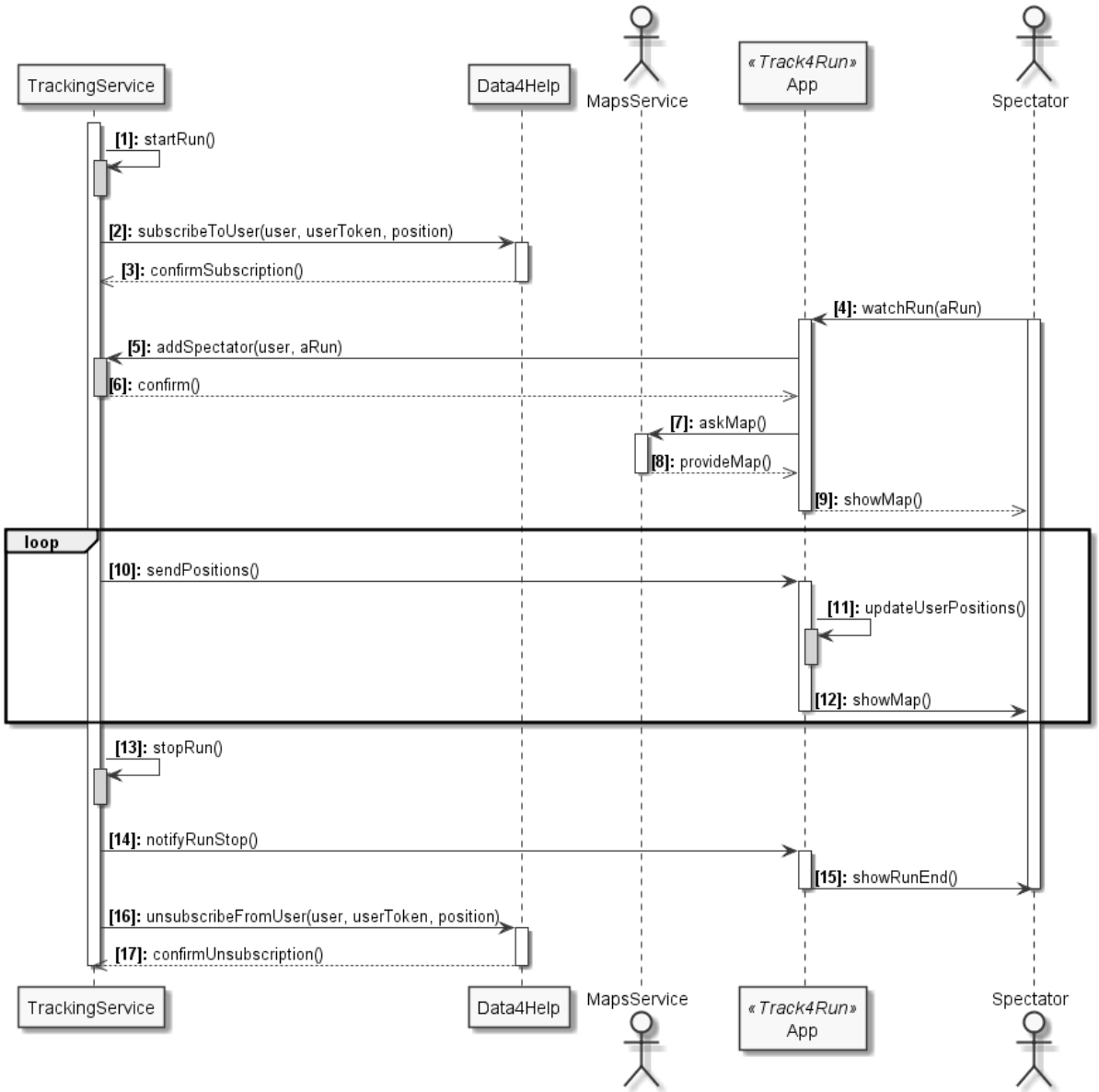


Figure 22: Track4Run - Spectator watching run sequence

3.3 Performance Requirements

In general, the main performance requirements for our system concern the time needed to process data and fulfill data requests.

In particular, the system should meet the following strong requirement:

- **AutomatedSOS Emergency Response** The system must call an ambulance within **5 seconds** from when a parameter exceeds a threshold. This time shall take into account the time needed for the AutomatedSOS monitoring system to process new incoming data and detect the emergency, the Data4Help system to send the new data to AutomatedSOS and the average delay introduced from the communication interfaces.

3.4 Design Constraints

3.4.1 Regulatory policies

- **GDPR**: In order to protect the user's privacy and perform a correct data treatment, the system must be developed in compliance with the latest GDPR regulations, giving the maximum attention to the protection and anonymization of the user's data.

3.4.2 Hardware limitations

The only hardware limitations of our system is present in the devices on which AutomatedSOS and Track4Run applications will be installed. For this reason, the development of the software to be will have to reasonably take in account some common limitations of these devices, such as power consumption and memory size.

3.5 Software System Attributes

3.5.1 Reliability

The system must be available 24/7. Since this requirement is quite demanding, small service breaks will be tolerated. To boost reliability, a RAID architecture that combines multiple disk drives to have data redundancy is suggested. Using a RAID controller we could also expand disks' capacity or substitute them without the risk of losing data or suspending the service for too long. Preventive maintenance is also a good idea to avoid downtimes.

3.5.2 Availability

To guarantee a 3-nines (99.9%) availability degree, a system of redundant servers could be considered. In this way we don't have a single point of failure and if one server fails, another one will be ready to substitute it.

3.5.3 Security

Security is a main issue. At first, to access the functionalities offered by our platform, users have to complete a login phase providing their credentials: fiscal code and password in case of a single user or VAT code and electronic signature for the third parties. These sensitive information should be confidentially stored and encrypted using a proper hash function.

Secondly, the system continuously collects data on health conditions from the users. Typically this kind of data are also considered sensitive and should be kept secret. Finally, also communications between users and our platform is very important, for this reason the system could be accessed by the customers using secure connection protocols like HTTPS to avoid Man In The Middle attacks. Only the

system administrator who is responsible of the Web and Application Server configuration can decide if it is the case to implement them or not.

3.5.4 Maintainability

In order to make our software the most maintainable as possible, we will adopt a modular design for our code. In this different components of the system can be modified and additional ones can be introduced in the system with whenever is needed and with a minimal impact on the rest of the system.

Moreover, we will make sure to adopt standard design patterns and coding best practices, so that our code can be easily understood and modified by any future developer.

Finally, the code should be completely and properly documented in order to facilitate the understanding of it.

3.5.5 Portability

During the system installation and setup, we have to consider different factors:

- Ease of installation on the central server
- Scalability of the platform considering future adjustments without losing the already collected data
- Portability of data between different machines, in order to move the system on several more powerful machines in case of necessity up 24/7

4 Formal Analysis Using Alloy

Here the complete alloy model is described.

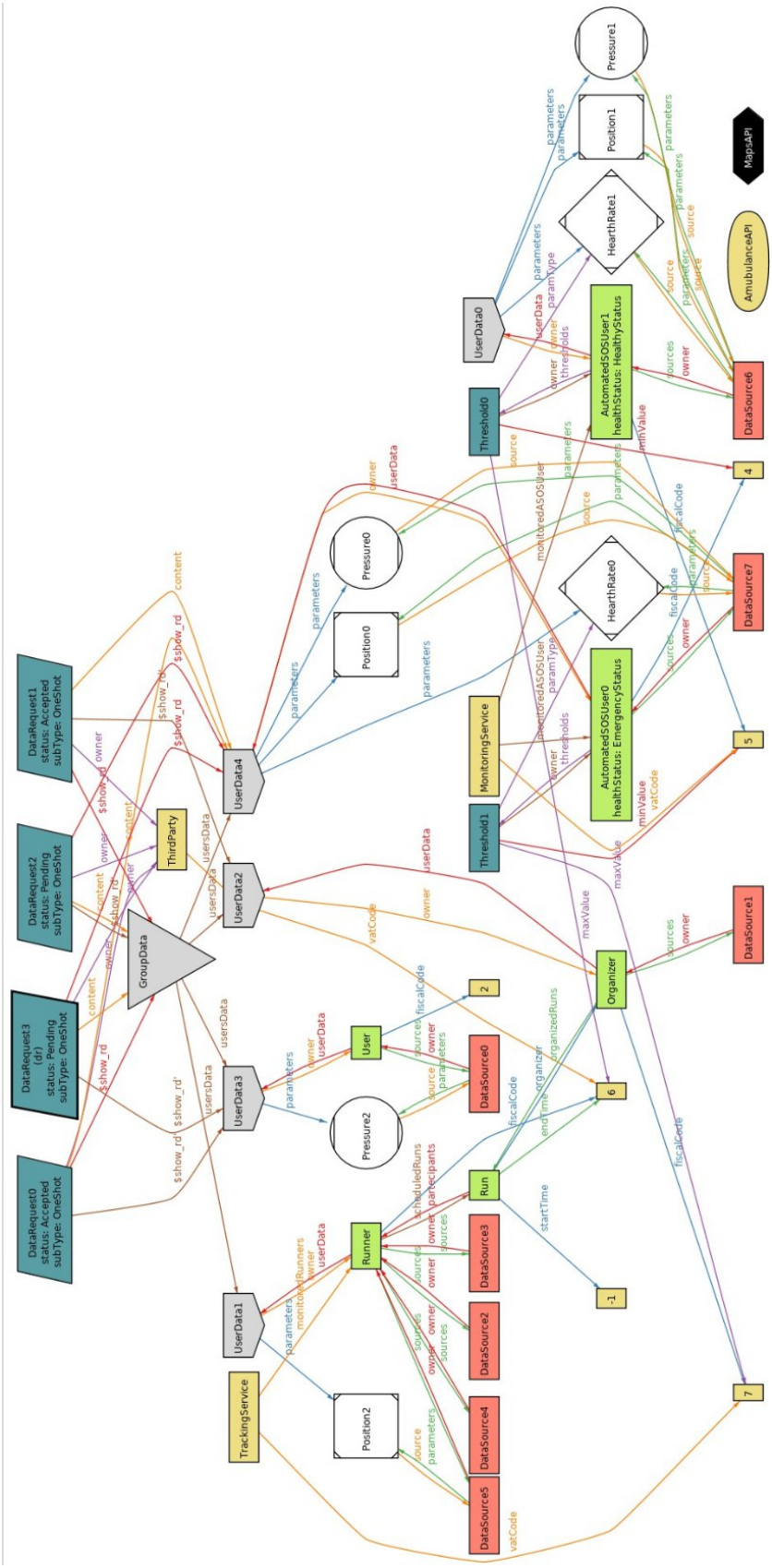
```
/** Signatures **/

-- Fiscal Code should be a string, here it's an int to be able to do operation on it (not
    ↪ permitted on string type in alloy)
sig User {
fiscalCode: one Int,
userData: one UserData,
sources: some DataSource
}
sig UserData extends RequestableData {
owner: one User,
parameters: set Parameter
}
sig DataSource {
owner: one User,
parameters: set Parameter
}
-- values of the Parameters are here commented because there isn't any check on them in
    ↪ the model
abstract sig Parameter {
source: one DataSource
}
sig Position extends Parameter {
--latitude: one Int,
--longitude: one Int
}
sig HearthRate extends Parameter {
--value: one Int
}
sig Pressure extends Parameter {
--value: one Int
}
sig ThirdParty {
vatCode: one Int,
}
sig DataRequest {
owner: one ThirdParty,
subType: one SubscriptionType,
status: one Status,
content: one RequestableData
}
abstract sig SubscriptionType {}
one sig OneShot extends SubscriptionType {}
one sig Subscription extends SubscriptionType {}

abstract sig Status {}
one sig Accepted extends Status {}
one sig Refused extends Status {}
one sig Pending extends Status {}

abstract sig HealthStatus {}
one sig HealthyStatus extends HealthStatus {}
one sig EmergencyStatus extends HealthStatus {}

-- RequestableData is the content of a thirdParty request, that can be for a single
    ↪ UserData or for GroupData, composed of many UserData
abstract sig RequestableData {
}
sig GroupData extends RequestableData {
usersData: some UserData
}
/* AutomatedSOS signatures*/

sig AutomatedSOSUser extends User {
healthStatus: one HealthStatus,
thresholds: some Threshold
}
sig Threshold {
```

```
owner: one AutomatedSOSUser,
paramType: one Parameter,
minValue: one Int,
maxValue: one Int
} {minValue < maxValue}
one sig MonitoringService extends ThirdParty {
monitoredASOSUser: set AutomatedSOSUser
}
one sig AmubulanceAPI {
}
/* Track4Run signatures*/

sig Runner extends User {
scheduledRuns: some Run
}
sig Organizer extends User {
organizedRuns: some Run
}
sig Spectator extends User {
}
sig Run {
organizer: one Organizer,
partecipants: set Runner,
startTime: one Int,
endTime: one Int
}
one sig TrackingService extends ThirdParty {
monitoredRunners: set Runner
}
one sig MapsAPI {
--connected to the TrackingService
}
/** Data4Help Facts **/

fact noEqualFiscalCode {
no disj u1,u2: User |  (u1.fiscalCode = u2.fiscalCode)
}
fact noEqualVATcode {
no disj t1,t2: ThirdParty | (t1.vatCode = t2.vatCode)
}
-- Over 1000 data makes the request successfull
fact groupDataExistence {
all g: GroupData | #g.usersData > 1  -- This should be 1000 but it's unfeasible to verify
    ↪  our model in a scope that large
}
-- Consistency of signatures connections
fact UserDataOwnership {
all u: User , ud: UserData | (ud.owner = u iff u.userData = ud )
}
fact UserSourcesOwnership {
all u: User , s: DataSource | ( (s in u.sources) implies s.owner = u ) and ( s.owner = u
    ↪ implies ( s in u.sources ) )
}
fact SourceParamOwnership {
all s: DataSource, p: Parameter | ( (p in s.parameters) implies p.source = s) and ( p.
    ↪ source = s implies (p in s.parameters))
}
-- All parameters tracked by a User's sources are present in the user's userData
fact userDataContainsAllSourcesParameters {
all ud: UserData, u: User, s: DataSource, p: Parameter | (ud.owner = u and (s in u.
    ↪ sources) and (p in s.parameters)) implies (p in ud.parameters)
}
-- Only parameters tracked by a User's sources are present in the user's userData
fact userDataContainsOnlyUserSourceParam {
all ud: UserData, u: User, s: DataSource, p: Parameter | ((ud.owner = u) and (not s in u.
    ↪ sources)) implies (( p in s.parameters) implies (not p in ud.parameters))
}
-- For each Parameter Type a declaration that every userData has at max one instance of
    ↪ it
fact userDataMaxOnePosition {
all ud: UserData | no disj p1,p2: Position | ((p1 in ud.parameters) and (p2 in ud.
    ↪ parameters))
```

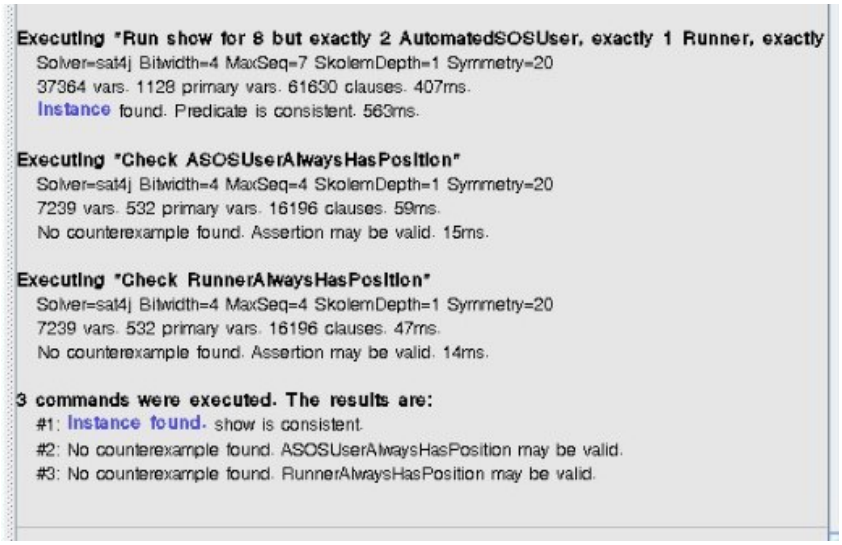
```
}
fact userDataMaxOneHeartRate {
all ud: UserData | no disj hb1,hb2: HearthRate | ((hb1 in ud.parameters) and (hb2 in ud.
    ↪ parameters))
}
fact userDataMaxOnePressure {
all ud: UserData | no disj bp1,bp2: Pressure | ((bp1 in ud.parameters) and (bp2 in ud.
    ↪ parameters))
}
/** AutomatedSOS Facts **/

-- All AutomatedSOSUsers are monitored by the MonitoringService
fact ASOSUinMonitoredList {
all au: AutomatedSOSUser | au in MonitoringService.monitoredASOSUser
}
-- MonitoringService never makes request not for UserData of a ASOS user
fact noOtherReqMonitoring {
no dr: DataRequest | all rd: RequestableData | (dr.owner = MonitoringService) and (dr.
    ↪ content = rd) and ( rd in GroupData or rd.owner not in AutomatedSOSUser)
}
-- Consistency of signatures connection
fact autoSOSuserThresholdOwnership {
all u: AutomatedSOSUser | all t: Threshold | (t in u.thresholds) iff (t.owner = u)
}
-- All request of MonitoringService are Accepted and Subscribed
fact monitoringServiceRequestAreSubAccSingle {
all r: DataRequest | one m: MonitoringService | r.owner = m implies ( (r.status in
    ↪ Accepted) and (r.subType in Subscription) )
}
-- Each AutomatedSOSUser has a DataSource that has Position as a parameter (location is
    ↪ needed to provide AutomatedSOS services)
fact ASOSUserNeedsLocation {
all u: AutomatedSOSUser | one s: DataSource | one p: Position | ( s in u.sources ) and (
    ↪ p in s.parameters )
}
-- Only Param of AutomatedSOSUser can have a threshold
fact noThresholdForNonASOSUser {
all t: Threshold | one au: AutomatedSOSUser | all s: DataSource | ( (s in au.sources) and
    ↪  (au.thresholds = t) ) iff ( t.paramType in s.parameters)
}

-- Threshold aren't applied to the Position
fact noThresholdPosition {
all t: Threshold | no p: Position | t.paramType = p
}
-- No more than one Threshold for each type of Param
fact noMultipleThresholdSameParam {
no disj t1,t2: Threshold | no hr: HearthRate | no p: Pressure | ( (t1.paramType = p) and
    ↪ (t2.paramType = p) ) and ( (t1.paramType = hr) and (t2.paramType = hr) )
}

/** Track4Run facts **/

-- All Runners are tracked by the TrackingService
fact runnerInTrackerList {
all r: Runner | r in TrackingService.monitoredRunners
}
-- TrackingService never makes request not for UserData of a Runner
fact noOtherReqTracking {
no dr: DataRequest | all rd: RequestableData | (dr.owner = TrackingService) and (dr.
    ↪ content = rd) and ( rd in GroupData or rd.owner not in Runner)
}
-- All request of TrackingService are Accepted and Subscribed
fact trackingServiceRequestAreSubAccSingle {
all r: DataRequest | one t: TrackingService | r.owner = t implies ( (r.status in Accepted
    ↪ ) and (r.subType in Subscription) )
}
-- Consistency of signatures connection
fact runnerEnrolledRunConnection {
all rer: Runner | all r: Run | ( r in rer.scheduledRuns) iff ( rer in r.partecipants)
}
fact organizerRunOwnership {
```

```
all o: Organizer | all r: Run | ( r in o.organizedRuns) iff ( o = r.organizer )
}
-- Each Runner has a DataSource that has Position as a parameter (location is needed to
    ↪ provide Track4Run services)
fact RunnerNeedsLocation {
all u: Runner | one s: DataSource | one p: Position | ( s in u.sources ) and ( p in s.
    ↪ parameters )
}
-- No runner can be enrolled to multiples runs that appens at the same time
fact noMultipleRunEnrollmentAtSameTime {
all r: Runner | no disj r1,r2: Run | -- all the possibility for overlapping runs
( ( r1.startTime < r2.startTime and r1.endTime > r2.endTime ) or ( r2.startTime < r1.
    ↪ startTime and r1.startTime < r2.endTime ) ) and
( r1 in r.scheduledRuns ) and ( r2 in r.scheduledRuns )
}

pred show {
(one u:User | #u.sources ≥2) and
(some dr: DataRequest | dr.content = GroupData)
}
assert ASOSUserAlwaysHasPosition {
all au: AutomatedSOSUser | one p: Position | p in au.userData.parameters
}
assert RunnerAlwaysHasPosition {
all r: Runner | one p: Position | p in r.userData.parameters
}

run show for 8 but exactly 2 AutomatedSOSUser, exactly 1 Runner, exactly 1 Run, exactly 1
    ↪  Organizer, exactly 3 ThirdParty, exactly 5 User, exactly 4 DataRequest

check ASOSUserAlwaysHasPosition
check RunnerAlwaysHasPosition
```

4.1 World Generated



4.2 Proof of consistency



5 Effort Spent

5.1 Andrea Biscontini

- General brainstorming : 5h
- Requirements bainstorming: 7h
- Alloy model : 10h
- Final review : 15h

5.2 Alvise de' Faveri Tron

- General brainstorming : 5h
- Requirements bainstorming: 7h
- UML models : 8h
- Final review : 15h

5.3 Marco Gelli

- General brainstorming : 5h
- Requirements bainstorming: 7h
- UI mockups : 9h
- Final review : 15h